Under consideration for publication in Formal Aspects of Computing

Modular reasoning in Z: scrutinising monotonicity and refinement

Moshe Deutsch[1], Martin C. Henson[1] and Steve Reeves[2]

[1]Department of Computer Science, University of Essex, Wivenhoe Park, Colchester, Essex CO4 3SQ, UK
[2]Department of Computer Science, University of Waikato, Private Bag 3105, Hamilton, New Zealand
E-mail: {mdeuts, hensm}@essex.ac.uk, stever@cs.waikato.ac.nz

Abstract. The schema calculus operators of Z provide an excellent means for expressing modular specifications but not for undertaking modular reasoning: it is well-known that these operators have poor monotonicity properties. The paper addresses three topics in this context: first, we provide a thorough mathematical analysis of monotonicity with respect to four schema operations and for three notions of operation refinement. Second, we provide a comprehensive analysis of the relational completion operator, known as *lifted-totalisation*, that underlies the standard notion of refinement in Z. Third, we provide a new semantics which induces a fully monotonic schema calculus.

Keywords: Monotonicity; Operation Refinement; Schema Calculus; Specification Language; Specification Logic

"We used to think that if we knew one, we knew two, because one and one are two. We are finding that we must learn a great deal more about 'and'." Sir Arthur Eddington (1882-1944)

1. Introduction

This paper addresses three topics:

1. It provides a systematic analysis of the monotonicity properties of four major operators taken from the schema calculus of Z, with respect to the standard notion of (operation) refinement. Failure of monotonicity is well-known, though there is precious little in the way of a formal analysis in the existing literature ([29, 18, 19] are recent exceptions). We isolate conditions under which monotonicity holds;

2. It provides a detailed analysis of the *lifted-totalisation* relational completion operation for partial relations. The standard model for Z is a partial relation semantics; the standard account of refinement is a total correctness interpretation. The lifted-totalisation completion mediates between the underlying semantics and the interpretation of refinement;

3. It establishes an alternative relational semantics for Z which leads to a fully monotonic schema calculus with respect to refinement.

In the remainder of this introduction we discuss the background and context of our investigations and outline the structure and organisation of the technical sections which follow.

Correspondence and offprint requests to: Martin C. Henson, Department of Computer Science, University of Essex, Wivenhoe Park, Colchester, Essex CO4 3SQ, UK.

1.1. Modular development in Z?

Z [56] is a specification language which permits the modular construction of specifications by means of an algebra of connectives and quantifiers reminiscent of predicate logic. This provides great structural expressivity and is a major reason for the popularity of the approach. Practical examples are covered in the better textbooks (*e.g.* [4], [65]) and its logical properties, justifying the usual terminology: schema *calculus*, in [36].
In addition to its use purely as a language for specification, there has been considerable interest in using Z for design and development. An approach which integrates work in refinement with Z is given in, for example, [13, 65, 51, 24]. Yet, a number of issues make this more general use for Z problematic, the most central being the fact that the modular techniques for expressing structured specifications is not accompanied by the possibility of modular *reasoning* because the schema operations are not monotonic with respect to the standard account of refinement. Thus, refinement takes place on operations expressed as a single schema: the schema calculus operators are removed (essentially by using an equational logic) before applying refinement (there are many examples and case studies, *e.g.* [61, 43, 42]).
As a consequence, many authors have developed case studies in Z without employing refinement at all (*e.g.* [49, 53, 31]) or hoping that their work will underlie software development using verification techniques (*e.g.* [38]). Other authors have addressed this by proposing methods for transforming Z specifications into fully monotonic frameworks such as Morgan's Refinement Calculus [48, 3] (*e.g.* [42, 63, 9, 11, 26]) or other notations based on Dijkstra's guarded command language [23] (*e.g.* [67, 43][1]). Unfortunately, these require either the elimination of the schema operators prior to the process or the utilisation of very strong sideconditions. Some solved this difficulty by introducing informal techniques for transforming Z specifications into declarative programming languages such as Haskell (*e.g.* [39, 27]). Others advocate substituting Z with a more powerful version of Refinement Calculus, which enriches the language with Z-like specification constructors in order to equip it with modularity capabilities (*e.g.* [58, 44, 28]); ironically, the major disadvantage of this method is that, in many cases, the additional specification constructors re-introduce non-monotonicity. A radically different approach is taken in [33] and [37], where the semantics of both operation schemas (but not state schemas) and the schema calculus is modified in order to attain a language that is both modular and fully monotonic with respect to refinement. This yields a Z-like system in which, however, the schema operators no longer express exactly their usual informal semantics. Unlike the other methods discussed above, this work takes place entirely within the specification language and its logic.

1.2. The partial relation semantics and equational logic of Z

In Z, schemas generally denote *partial* relations. For example, consider the schema:

$$Predecessor \mathrel{\widehat=} [\, \mathtt{x}, \mathtt{x}' : \mathbb{N} \mid \mathtt{x}' = \mathtt{x} - 1 \,]$$

The set this denotes contains *bindings* (valid assignments of values to the observations $\mathtt{x}$ and $\mathtt{x}'$) of the form:

$$\langle\!|\, \mathtt{x} \Rrightarrow n, \mathtt{x}' \Rrightarrow n - 1 \,|\!\rangle$$

for all $n > 0$. But no binding of the form:

$$\langle\!|\, \mathtt{x} \Rrightarrow 0, \mathtt{x}' \Rrightarrow m \,|\!\rangle$$

for any $m \in \mathbb{N}$. In this sense the schema is partial when viewed as a relation[2] between its *before-state* (the sub-binding involving the observation $\mathtt{x}$) and *after-state* (the sub-binding involving the observation $\mathtt{x}'$).
The underlying relation for schemas in Z is not *refinement* but *equality*. The definition of the schema operators leads directly to an *equational logic*. One such equation, characterising conjunction, is:

$$[\, D_0 \mid P_0 \,] \wedge [\, D_1 \mid P_1 \,] = [\, D_0;\ D_1 \mid P_0 \wedge P_1 \,]$$

There are similar equations for all schema operators. By orienting these equalities, it is easy to see that *every* schema expression is trivially equal to a single atomic schema (its normal form, so to speak).
What behaviour is permitted for a correct implementation of a (partial) specification outside the domain of the relation it denotes? To answer this question we need a theory of *refinement*: a means for comparing such an implementation

[1] These are based on a method, developed by J. B. Wordsworth, used by IBM UK laboratories at Hursley.
[2] We will use U (*etc.*) in future to range over schema expressions. Strictly speaking these are *interpreted* as partial relations in the underlying theory $\mathcal{Z}_C$ (see [36]) or $\mathcal{Z}_C^{\perp}$ (see [21] and a summary in appendix A). Only from section 5.1 will it be necessary to distinguish between them, so until then, for simplicity of presentation, we will write U both for a schema expression (syntax) *and* for the partial relation (semantics) it denotes.

with such a specification. The standard notion of refinement in Z is based on a subsequent *total-relation semantics*, known as the *lifted-totalised interpretation* [65, 13]. Under this interpretation the answer to the question is: *anything can happen* (this is sometimes known as *chaotic behaviour*). This is not the only theory of refinement which can be developed on the basis of the underlying partial relation semantics. However, as we shall see in detail, all the various options lead to refinement theories for which, to varying degrees, monotonicity of the schema operators fails.

1.3. Organisation and overview

The paper begins, in section 2, with an overview of operation refinement. We then move on, in section 3, to an analysis of the monotonicity properties of four standard schema calculus operators, each of which is investigated with respect to three notions of operation refinement in Z. Unlike [33] and [37], we pursue our investigation in Z as it is informally understood in, for example, [54] and [65]. A related analysis for schema conjunction and schema disjunction was presented in [29]. We explore those operators in detail and, additionally, schema existential hiding and schema composition. The investigations suggest a number of *sideconditions* that can be used as "healthiness conditions" for Z specifications, which guarantee monotonicity. The usefulness of these sideconditions is also discussed.
In section 4, the second part of the paper, we investigate the *distributivity properties* of the lifted-totalisation operator with respect to the four schema operations. We show that none of these operators fully distributes over the lifted-totalisation, uncover reasons for this failure and introduce sideconditions for full-distribution in each case.
These investigations motivate the final part of the paper (in section 5): the introduction of a fully monotonic schema calculus based on the lifted-totalised interpretation as the underlying semantics for atomic schemas. The semantics for the schema operations is then given recursively, using the standard operations, and refinement merely becomes the subset relation on the semantics.
Our investigations become possible in virtue of the logic for Z reported in, for example, [36] and benefit from the technique of rendering all the theories of refinement in a proof-theoretic form: as sets of introduction and elimination rules. This leads to a uniform and simple method for proving the various technical results.
We provide some essential notational conventions, and a complete formal account of the theory of preconditions for compound operations, in appendix A;[3] this is an important precursor for the investigations presented in sections 3 and 4. The paper concludes with a summary and an agenda for further investigation (section 6).

2. Operation refinement

Operation refinement concerns the derivation of a more concrete operation from a given abstract one, without changing the specification of the underlying state. It is sometimes called *algorithm design* [67]. The partial relation semantics of operation schemas in Z raises an immediate question: what does it mean for one operation schema to refine another? More generally: what does it mean for one partial relation to refine another? The standard answer involves some sort of lifted-totalisation of the underlying partial relations (see *e.g.* [65] and [13]). In [21] it is shown that the relational completion notion is equivalent to various other approaches, one of which, S-refinement (and its relatives), is much simpler and more intuitive. Therefore, for the analysis of monotonicity (section 3) we shall work entirely within the simpler S-refinement framework. This permits us to deal directly with partial, rather than with lifted-totalised, relations and, at least at this stage, to avoid the complications involving additional semantic elements.
We begin by introducing three distinct notions of operation refinement in Z, based on three distinct answers to the questions above.

2.1. S-refinement

We introduce a pure proof-theoretic characterisation of refinement, which is closely connected to sufficient refinement conditions introduced by Spivey (hence "S"-refinement) in [54] and as discussed in, for example, [42, 51, 67, 65].
This notion is based on two basic observations regarding the properties one expects in a refinement: first, that a refinement may involve the reduction of nondeterminism; second, that it may also involve the expansion of the domain of definition. Put another way, we have a refinement providing that *postconditions do not weaken* and that *preconditions do not strengthen*.

[3] This is sufficient for our needs in this paper. Further detail can be found in [36] and [21].

This notion can be captured by forcing the refinement relation to hold *exactly* when these conditions apply. S-refinement is written $U_0 \sqsupseteq_s U_1$ (U_0 S-refines U_1) and is given by the definition that leads directly to the following rules:

Proposition 2.1. Let z, z_0, z_1 be fresh variables.

$$\frac{Pre\ U_1\ z \vdash Pre\ U_0\ z \quad Pre\ U_1\ z_0, z_0 \star z_1' \in U_0 \vdash z_0 \star z_1' \in U_1}{U_0 \sqsupseteq_s U_1}\ (\sqsupseteq_s^+)$$

$$\frac{U_0 \sqsupseteq_s U_1 \quad Pre\ U_1\ t}{Pre\ U_0\ t}\ (\sqsupseteq_{s_0}^-) \qquad \frac{U_0 \sqsupseteq_s U_1 \quad Pre\ U_1\ t_0 \quad t_0 \star t_1' \in U_0}{t_0 \star t_1' \in U_1}\ (\sqsupseteq_{s_1}^-)$$

□

We prove in [21] and [20] that S-refinement is equivalent to several other characterisations of refinement, such as W$_\bullet$-refinement based on Woodcock's *chaotic* relational completion model [65, 13] (see appendix A, definition A.10). As we remarked above, S-refinement deals directly with the partial relation semantics rather than by means of an interpretation as a (lifted-totalised) relation. It is, therefore, simpler both as a theory and in terms of the analysis we undertake in section 3.

2.2. SP-refinement

This is an alternative proof-theoretic characterisation of refinement, which is closely connected to refinement in the *behavioural* [13] or *firing condition* [55] approach. This special case of S-refinement may involve reduction of non-determinism but insists on the *stability of the precondition*. SP-refinement is written $U_0 \sqsupseteq_{sp} U_1$ and is given by the definition that leads directly to the following rules:

Proposition 2.2. Let z, z_0, z_1 be fresh variables.

$$\frac{Pre\ U_1\ z \vdash Pre\ U_0\ z \quad z_0 \star z_1' \in U_0 \vdash z_0 \star z_1' \in U_1}{U_0 \sqsupseteq_{sp} U_1}\ (\sqsupseteq_{sp}^+)$$

$$\frac{U_0 \sqsupseteq_{sp} U_1 \quad Pre\ U_1\ t}{Pre\ U_0\ t}\ (\sqsupseteq_{sp_0}^-) \qquad \frac{U_0 \sqsupseteq_{sp} U_1 \quad t_0 \star t_1' \in U_0}{t_0 \star t_1' \in U_1}\ (\sqsupseteq_{sp_1}^-)$$

□

We showed in [20] that SP-refinement is equivalent to several other characterisations of refinement. For example, W$_\Box$-refinement, which is based on the *abortive* relational completion model (see appendix A, definition A.12) and discussed in [6] and [13].

2.3. SC-refinement

SC-refinement is our third alternative proof-theoretic characterisation of refinement. It is written $U_0 \sqsupseteq_{sc} U_1$ and is given by the definition that leads directly to the following rules:

Proposition 2.3. Let z_0, z_1 be fresh variables

$$\frac{z_0 \star z_1' \in U_1 \vdash z_0 \star z_1' \in U_0 \quad Pre\ U_1\ z_0, z_0 \star z_1' \in U_0 \vdash z_0 \star z_1' \in U_1}{U_0 \sqsupseteq_{sc} U_1}\ (\sqsupseteq_{sc}^+)$$

$$\frac{U_0 \sqsupseteq_{sc} U_1 \quad t_0 \star t_1' \in U_1}{t_0 \star t_1' \in U_0}\ (\sqsupseteq_{sc_0}^-) \qquad \frac{U_0 \sqsupseteq_{sc} U_1 \quad Pre\ U_1\ t_0 \quad t_0 \star t_1' \in U_0}{t_0 \star t_1' \in U_1}\ (\sqsupseteq_{sc_1}^-)$$

□

Lemma 2.1. The following extra rule is derivable for SC-refinement:

$$\frac{U_0 \sqsupseteq_{sc} U_1 \quad Pre\ U_1\ t}{Pre\ U_0\ t}$$

□

SC-refinement is introduced for technical reasons which inform the analysis to follow. This notion, in which the precondition may weaken, but in which the postcondition is stable, is not otherwise of much pragmatic interest.

3. Operation refinement and monotonicity in the schema calculus

The major advantage of Z, in contrast to other paradigms such as the Refinement Calculus and even B [2], is its potential for expressing modular specifications using schema operators. However, in order to properly exploit modularity and in particular to undertake specification refinement, it is vital that the various schema operators of the language are monotonic. When monotonicity holds, the components of a composite specification can be refined independently of the remainder of the specification [29]; refinement can then be performed in a modular manner. Unfortunately it is well-known folk-lore that the Z schema calculus operators have very poor monotonicity properties; this has a major effect on their usefulness, for example, in the context of program development from Z specifications.
In this section we analyse the monotonicity properties of four of the most interesting schema calculus operators (conjunction, disjunction, existential hiding and composition) with respect to each one of the refinement theories presented in section 2. We provide examples of monotonicity (or non-monotonicity) and establish sideconditions, as "healthiness conditions" on specifications, in order to attain monotonicity. We also discuss the usefulness of these sideconditions in the context of the various refinement theories we consider.

3.1. Refinement for conjunction

We do not have an introduction rule for the precondition of conjoined operations (see appendix A, section A.3.1). Consequently, schema conjunction is *not monotonic* with respect to S-refinement.
Here is a simple counterexample. Consider the following schemas:

$$U_0 \mathrel{\widehat=} [\, \mathtt{x}, \mathtt{x}' : \mathbb{N} \mid \mathtt{x}' = 8 \,] \quad U_1 \mathrel{\widehat=} [\, \mathtt{x}, \mathtt{x}' : \mathbb{N} \mid \mathtt{x}' < 10 \,] \quad U_2 \mathrel{\widehat=} [\, \mathtt{x}, \mathtt{x}' : \mathbb{N} \mid \mathtt{x}' = 2 \,]$$

We note that U_1 is a nondeterministic operation that can be refined by strengthening its postcondition, for example to U_0. However, when conjoining the operations, we have the following schemas:

$$U_0 \wedge U_2 = [\, \mathtt{x}, \mathtt{x}' : \mathbb{N} \mid \mathit{false} \,] \quad U_1 \wedge U_2 = [\, \mathtt{x}, \mathtt{x}' : \mathbb{N} \mid \mathtt{x}' = 2 \,]$$

In [21] and [20], we define the chaotic specification as: $Chaos =_{df} [T \mid \mathit{false}]$. A chaotic specification cannot constitute a refinement of any other specification because this would signify *augmentation of undefinedness* and would therefore violate any notion of refinement presented in section 2.[4] Thus, $U_0 \wedge U_2 \not\sqsupseteq_s U_1 \wedge U_2$.
The counterexample also shows the reason for the failure: strengthening the postcondition might create a chaotic specification, due to both the "postcondition only" (single predicate) approach Z takes [47, 60, 59], and the definition of schema conjunction. This motivates the following sidecondition. It is perhaps not surprising that it is precisely the missing introduction rule for the precondition of conjoined operations.

Proposition 3.1. Let z be a fresh variable and U_0, U_1, U_2 be operation schemas with the property that:

$$\mathit{Pre}\ U_0\ z \wedge \mathit{Pre}\ U_2\ z \Rightarrow \mathit{Pre}\,(U_0 \wedge U_2)\ z$$

Then the following rule is derivable:

$$\frac{U_0 \sqsupseteq_s U_1}{U_0 \wedge U_2 \sqsupseteq_s U_1 \wedge U_2}$$

[4] See [21, section 4.4] for further details.

Proof

$$\dfrac{\dfrac{U_0 \sqsupseteq_s U_1 \quad \dfrac{\overline{Pre\,(U_1 \wedge U_2)\,z}\ (1)}{Pre\ U_1\ z}}{Pre\ U_0\ z} \qquad \dfrac{\overline{Pre\,(U_1 \wedge U_2)\,z}\ (1)}{Pre\ U_2\ z}}{\dfrac{Pre\ U_0\ z \wedge Pre\ U_2\ z}{\dfrac{Pre\,(U_0 \wedge U_2)\,z \qquad z_0 \star z_1' \in U_1 \wedge U_2\ \delta}{U_0 \wedge U_2 \sqsupseteq_s U_1 \wedge U_2}\ (1)}}$$

Where δ is:

$$\dfrac{\dfrac{U_0 \sqsupseteq_s U_1 \quad \dfrac{\overline{Pre\,(U_1 \wedge U_2)\,z_0}\ (1)}{Pre\ U_1\ z_0}}{z_0 \star z_1' \mathrel{\dot\in} U_1} \qquad \dfrac{\dfrac{\overline{z_0 \star z_1' \in U_0 \wedge U_2}\ (1)}{z_0 \star z_1' \mathrel{\dot\in} U_0} \qquad \dfrac{\overline{z_0 \star z_1' \in U_0 \wedge U_2}\ (1)}{z_0 \star z_1' \mathrel{\dot\in} U_2}}{z_0 \star z_1' \in U_1 \wedge U_2}$$

□

Much the same observation can be made for SP-refinement: since non-monotonicity follows by a permissible reduction of nondeterminism, SP-refinement and S-refinement coincide. Proposition 3.1 with $\sqsupseteq_s$ substituted by $\sqsupseteq_{sp}$ holds for SP-refinement; the proof is similar.
SC-refinement guarantees that no reduction of nondeterminism is possible, thus ensuring that schema conjunction is monotonic with respect to SC-refinement.

Proposition 3.2. The following rule is derivable:

$$\dfrac{U_0 \sqsupseteq_{sc} U_1}{U_0 \wedge U_2 \sqsupseteq_{sc} U_1 \wedge U_2}$$

Proof

$$\dfrac{\dfrac{U_0 \sqsupseteq_{sc} U_1 \quad \dfrac{\overline{z_0 \star z_1' \in U_1 \wedge U_2}\ (1)}{z_0 \star z_1' \mathrel{\dot\in} U_1}}{z_0 \star z_1' \mathrel{\dot\in} U_0} \qquad \dfrac{\overline{z_0 \star z_1' \in U_1 \wedge U_2}\ (1)}{z_0 \star z_1' \mathrel{\dot\in} U_2}}{\dfrac{z_0 \star z_1' \in U_0 \wedge U_2 \qquad z_0 \star z_1' \in U_1 \wedge U_2\ \delta}{U_0 \wedge U_2 \sqsupseteq_{sc} U_1 \wedge U_2}\ (1)}$$

Where δ is the δ branch of the proof of proposition 3.1 (with $\sqsupseteq_s$ substituted by $\sqsupseteq_{sc}$). □

Note that, although the non-monotonicity of conjunction with respect to S-refinement and SP-refinement is a direct consequence of the ability to strengthen the postcondition, the sidecondition used in proposition 3.1 is applied in the proof branch concerning the precondition. This is not surprising because, if one attempts to prove the refinement of the conjoined schemas given in the counterexample, one discovers that the branch for the postcondition is provable, due to $false$ in the antecedent of the implication; whereas the branch for the precondition fails for the opposite reason ($false$ in the consequent). Thus, one can expect an application of the sidecondition in this branch at some point.
We would like to highlight the value of insights gained from both counterexamples and the formal proofs. One of the benefits of a precise investigation is the ability to deduce or motivate various results as a direct consequence of these. The sidecondition above can be calculated through the direct attempt to prove proposition 3.1.

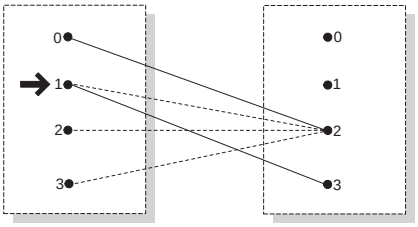

Fig. 1. The solid lines represent the partial relation denoted by the schema $U_1 \lor U_2$, and the dotted lines represent the additional behaviours in the schema $U_0 \lor U_2$. Note the point (marked with a right arrow) which represents *weakening of the postcondition* with respect to $U_1 \lor U_2$. Hence $U_0 \lor U_2 \not\sqsupseteq_s U_1 \lor U_2$.

3.2. Refinement for disjunction

Schema disjunction is *not monotonic* with respect to S-refinement. In contrast to the analysis of schema conjunction in section 3.1, the reason for non-monotonicity in this case is the fact that S-refinement enables us to weaken the precondition. Weakening the precondition of (at least) one component specification might extend the domain of the disjunction of the two components, leading to an *increase in nondeterminism* and thus a failure of refinement.
For example, consider the following schemas:

$$U_0 \mathrel{\widehat=} [\, \mathbf{x}, \mathbf{x}' : \mathbb{N} \mid \mathbf{x}' = 2 \,] \qquad U_1 \mathrel{\widehat=} [\, \mathbf{x}, \mathbf{x}' : \mathbb{N} \mid \mathbf{x} = 0 \land \mathbf{x}' = 2 \,]$$

$$U_2 \mathrel{\widehat=} [\, \mathbf{x}, \mathbf{x}' : \mathbb{N} \mid \mathbf{x} = 1 \land \mathbf{x}' = 3 \,]$$

The specification U_1 is partial and, therefore, can be refined to U_0 by weakening its precondition. However, respectively disjoining the schemas above yields the following specifications:

$$U_0 \lor U_2 = [\, \mathbf{x}, \mathbf{x}' : \mathbb{N} \mid \mathbf{x}' = 2 \lor \mathbf{x} = 1 \land \mathbf{x}' = 3 \,]$$

$$U_1 \lor U_2 = [\, \mathbf{x}, \mathbf{x}' : \mathbb{N} \mid \mathbf{x} = 0 \land \mathbf{x}' = 2 \lor \mathbf{x} = 1 \land \mathbf{x}' = 3 \,]$$

Clearly $U_0 \lor U_2 \not\sqsupseteq_s U_1 \lor U_2$. This is because the schema $U_0 \lor U_2$ permits the behaviour $(\!|\, \mathbf{x} \Rrightarrow 1, \mathbf{x}' \Rrightarrow 2 \,|\!)$, which is prohibited by $U_1 \lor U_2$. This is a representative example: the only reason why S-refinement can fail in such a case is as a result of an augmentation of nondeterminism with respect to the abstract disjunction; this is shown in Fig. 1. The analysis suggests that the sidecondition will be required in the proof branch concerning the postcondition; as we will now see, this is the case.

Proposition 3.3. Let z be fresh and U_0, U_1, U_2 be operation schemas with the property that:

$$Pre\ U_0\ z \land Pre\ U_2\ z \Rightarrow Pre\ U_1\ z$$

Then the following rule is derivable:

$$\frac{U_0 \sqsupseteq_s U_1}{U_0 \lor U_2 \sqsupseteq_s U_1 \lor U_2}$$

Proof

$$\dfrac{\dfrac{}{Pre\,(U_1 \lor U_2)\ z}\ (1) \qquad \dfrac{\dfrac{U_0 \sqsupseteq_s U_1 \quad \dfrac{}{Pre\ U_1\ z}\ (2)}{Pre\ U_0\ z}}{Pre\,(U_0 \lor U_2)\ z} \qquad \dfrac{\dfrac{}{Pre\ U_2\ z}\ (2)}{Pre\,(U_0 \lor U_2)\ z}}{Pre\,(U_0 \lor U_2)\ z}\ (2) \qquad \begin{array}{c}\delta_0 \\ \vdots \\ z_0 \star z_1' \in U_1 \lor U_2\end{array}}{U_0 \lor U_2 \sqsupseteq_s U_1 \lor U_2}\ (1)$$

Where δ_0 stands for the following branch:

$$\dfrac{\dfrac{}{z_0 \star z_1' \in U_0 \lor U_2}\,(1)}{z_0 \star z_1' \in U_0 \lor U_2}\,(1) \qquad \dfrac{\dfrac{\dfrac{}{z_0 \star z_1' \in U_0 \lor U_2}\,(1)}{z_0 \star z_1' \in T^\star} \qquad \overset{\delta_1}{\vdots}\ z_0 \star z_1' \dot{\in} U_1}{z_0 \star z_1' \in U_1 \lor U_2} \qquad \dfrac{\dfrac{}{z_0 \star z_1' \in U_0 \lor U_2}\,(1)}{z_0 \star z_1' \in T^\star} \qquad \dfrac{}{z_0 \star z_1' \dot{\in} U_2}\,(3)}{z_0 \star z_1' \in U_1 \lor U_2}\,(3)$$

and δ_1 is:

$$\dfrac{U_0 \sqsupseteq_s U_1 \qquad \dfrac{\dfrac{}{Pre\,(U_1 \lor U_2)\,z_0}\,(1) \quad \dfrac{}{Pre\ U_1\ z_0}\,(4)}{Pre\ U_1\ z_0} \qquad \dfrac{\dfrac{\dfrac{}{z_0 \star z_1' \dot{\in} U_0}\,(3)}{Pre\ U_0\ z_0} \quad \dfrac{}{Pre\ U_2\ z_0}\,(4)}{Pre\ U_0\ z_0 \land Pre\ U_2\ z_0} \ \vdots\ Pre\ U_1\ z_0}\,(4) \qquad \dfrac{}{z_0 \star z_1' \dot{\in} U_0}\,(3)}{z_0 \star z_1' \dot{\in} U_1}$$

□

Once again, the sidecondition is determined by the proof: it is precisely the entailment of an otherwise unprovable proposition ($Pre\ U_1\ z_0$) from the available assumptions ($Pre\ U_0\ z_0$ and $Pre\ U_2\ z_0$).

The situation with SC-refinement is very similar since, as the counterexample illustrates, the critical factor leading to non-monotonicity in this case is weakening of the precondition. SC-refinement sanctions this and, therefore, proposition 3.3 holds with $\sqsupseteq_s$ substituted by $\sqsupseteq_{sc}$. The proof, given this substitution, is similar.

Schema disjunction is monotonic with respect to SP-refinement because weakening of the precondition is prohibited. The following rule is derivable:

Proposition 3.4.

$$\dfrac{U_0 \sqsupseteq_{sp} U_1}{U_0 \lor U_2 \sqsupseteq_{sp} U_1 \lor U_2}$$

Proof

$$\dfrac{\overset{\delta_0}{\vdots}\ Pre\,(U_0 \lor U_2)\,z \qquad \overset{\delta_1}{\vdots}\ z_0 \star z_1' \in U_1 \lor U_2}{U_0 \lor U_2 \sqsupseteq_{sp} U_1 \lor U_2}\,(1)$$

Where δ_0 is identical to the precondition branch in the proof of proposition 3.3 (with $\sqsupseteq_s$ substituted by $\sqsupseteq_{sp}$) and δ_1 stands for the following branch:

$$\dfrac{\dfrac{}{z_0 \star z_1' \in U_0 \lor U_2}\,(1)}{z_0 \star z_1' \in U_0 \lor U_2}\,(1) \qquad \dfrac{\dfrac{\dfrac{}{z_0 \star z_1' \in U_0 \lor U_2}\,(1)}{z_0 \star z_1' \in T^\star} \qquad \dfrac{U_0 \sqsupseteq_{sp} U_1 \qquad \dfrac{}{z_0 \star z_1' \dot{\in} U_0}\,(3)}{z_0 \star z_1' \dot{\in} U_1}}{z_0 \star z_1' \in U_1 \lor U_2} \qquad \dfrac{\dfrac{}{z_0 \star z_1' \in U_0 \lor U_2}\,(1)}{z_0 \star z_1' \in T^\star} \qquad \dfrac{}{z_0 \star z_1' \dot{\in} U_2}\,(3)}{z_0 \star z_1' \in U_1 \lor U_2}\,(3)$$

□

3.3. Refinement for existential quantification

There is, of course, an intimate relationship between disjunction and existential quantification. We might expect the monotonicity properties of schema existential quantification to be similar to those for schema disjunction. Indeed, schema existential quantification is *not monotonic* with respect to S-refinement because weakening of the precondition might admit behaviours in the concrete operation that are unacceptable in the abstract. The reason is that schema existential quantification can hide an arbitrary observation and, in particular, observations that can lead to an augmentation of nondeterminism. This is shown by the counterexample in Fig. 2.

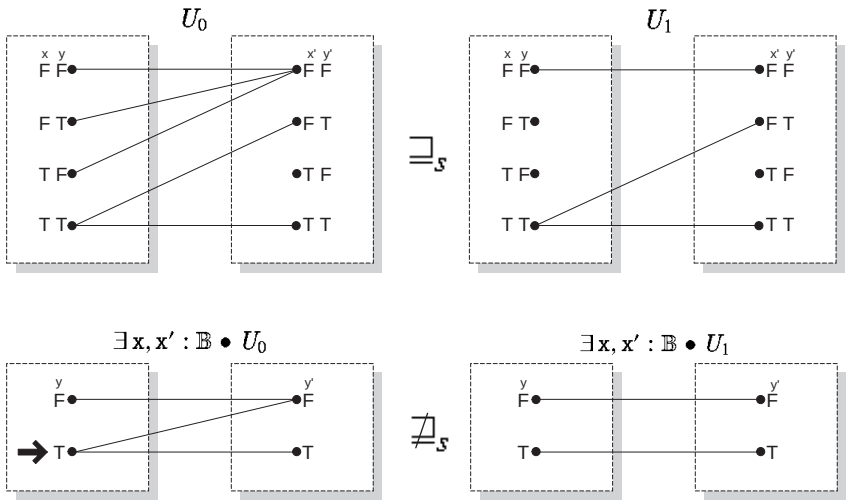

Fig. 2. A counterexample: schema existential quantification is not monotonic with respect to S-refinement.

We present the specifications U_0 and U_1, whose alphabets comprise the boolean observations $\mathtt{x}$, $\mathtt{x}'$, $\mathtt{y}$ and $\mathtt{y}'$. The specifications $\exists\, \mathtt{x}, \mathtt{x}' : \mathbb{B} \bullet U_0$ and $\exists\, \mathtt{x}, \mathtt{x}' : \mathbb{B} \bullet U_1$ hide the pair of observations $\mathtt{x}$ and $\mathtt{x}'$ from U_0 and U_1. Note that the schema U_1 denotes a partial operation and U_0 S-refines it by weakening the precondition. Nevertheless, hiding those observations introduces a weakening of the postcondition (marked with a right arrow in Fig. 2) of $\exists\, \mathtt{x}, \mathtt{x}' : \mathbb{B} \bullet U_0$ with respect to $\exists\, \mathtt{x}, \mathtt{x}' : \mathbb{B} \bullet U_1$. Hence, S-refinement fails.
Like schema disjunction, the above counterexample also suggests that, in order to prove monotonicity of existential hiding, a sidecondition is required in the proof branch concerning the postcondition. The sidecondition here is stronger than the one used for disjunction because, unlike schema disjunction, we do not have an additional disjoined schema.

Proposition 3.5. Let z be fresh and U_0, U_1 be operation schemas with the property that:

$$Pre\ U_0\ z \Rightarrow Pre\ U_1\ z$$

Then the following rule is derivable:

$$\frac{U_0 \sqsupseteq_s U_1}{\exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U_0 \sqsupseteq_s \exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U_1}$$

Proof

$$\dfrac{\dfrac{}{Pre\,(\exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U_1)\, z}\ (1) \qquad \dfrac{\dfrac{\dfrac{U_0 \sqsupseteq_s U_1 \quad \dfrac{}{Pre\ U_1\ y}\ (2)}{Pre\ U_0\ y}}{Pre\,(\exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U_0)\, y} \quad \dfrac{}{y \doteq z}\ (2)}{Pre\,(\exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U_0)\, z}\ (2) \qquad \begin{array}{c}\delta_0 \\ \vdots \\ z_0 \star z_1' \in \exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U_1\end{array}}{\exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U_0 \sqsupseteq_s \exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U_1}\ (1)$$

Where δ_0 is:

$$\dfrac{\dfrac{}{Pre\,(\exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U_1)\, z_0}\ (1) \qquad \dfrac{\dfrac{\dfrac{U_0 \sqsupseteq_s U_1 \quad \dfrac{}{Pre\ U_1\ w}\ (3)}{Pre\ U_0\ w}}{Pre\,(\exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U_0)\, w} \quad \dfrac{}{w \doteq z_0}\ (3)}{Pre\,(\exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U_0)\, z_0} \qquad \begin{array}{c}\delta_1 \\ \vdots \\ z_0 \star z_1' \in \exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U_1\end{array}}{z_0 \star z_1' \in \exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U_1}\ (4)}{z_0 \star z_1' \in \exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U_1}\ (3)$$

and δ_1 is:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{z_0 \star z_1' \in \exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U_0}\,(1)}{z_0 \star z_1' \in \exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U_1}\,(5)}{y_1 \mathrel{\dot\in} \exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U_1} \qquad \dfrac{}{y_1 \doteq z_0 \star z_1'}\,(5)}{y_1 \in U_1} \qquad \dfrac{}{y_1 \in U_0}\,(5)}{Pre\ U_0\ y_1} \qquad \vdots \qquad \dfrac{\dfrac{}{Pre\ U_0\ y_0}\,(4) \qquad \dfrac{}{y_0 \doteq z_0}\,(4) \quad \dfrac{}{y_1 \doteq z_0 \star z_1'}\,(5)}{y_0 \doteq y_1}}{U_0 \sqsupseteq_s U_1}$$

□

Note that the above sidecondition forces a "fixed precondition" refinement, which is precisely SP-refinement. So it is an immediate consequence that schema existential hiding is monotonic with respect to SP-refinement:

Proposition 3.6. The following rule is derivable:

$$\frac{U_0 \sqsupseteq_{sp} U_1}{\exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U_0 \sqsupseteq_{sp} \exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U_1}$$

□

The proof is essentially identical to proposition 3.5.

One might expect that the monotonicity properties of existential hiding with respect to S-refinement and SC-refinement would coincide. Indeed, the sidecondition of proposition 3.5 is required for proving monotonicity of schema existential hiding with respect to SC-refinement. However, since the sidecondition guarantees stability of the precondition, and SC-refinement the stability of the postcondition, the result holds only when the abstract and concrete operations are *equivalent*, which is of course, far from useful.

3.4. Refinement for composition

It is not surprising that schema composition is *not monotonic* with respect to S-refinement, because composition in Z can be expressed in terms of *conjunction* and *existential quantification* (appendix A, definition A.7(iii)). Consideration of the results of sections 3.1 and 3.3 suggests that both weakening the precondition and strengthening the postcondition of the component specifications will cause a problem. This is fairly intuitive since reduction of nondeterminism is *demonic* with respect to schema composition (since strengthening the postcondition on the *left* of the composition might demonically choose those after-states that do not connect with the precondition of the operation to the right of the composition). This results in losing requirements from the domain of the composed specifications. In addition, weakening the precondition on the *right* of the composition might demonically extend the specification domain in such a way that the composition will introduce unacceptable behaviours mapped from the original precondition.

The following counterexample illustrates the problems and motivates a solution by suggesting a sidecondition. Consider the following specifications:

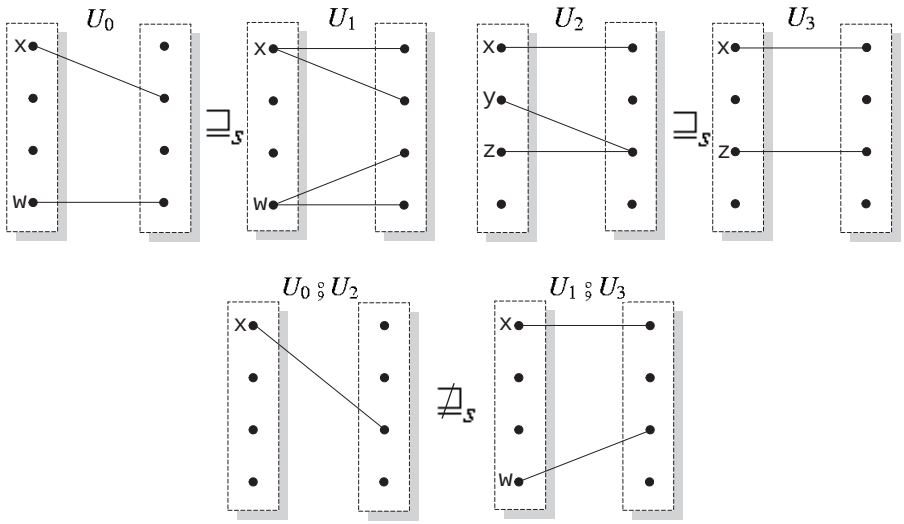

Fig. 3. A counterexample: schema composition is not monotonic with respect to refinement.

We introduce two abstract specifications, U_1 and U_3, and their respective S-refinements U_0 and U_2. We then show that the composition of the underlying concrete specifications does not constitute a refinement of the composition of their abstract counterparts. We label the before-states of each specification using the labels `x`, `y`, `z` and `w` (from the top).
The specification U_1 has two instances of nondeterminism and U_0 S-refines it by reducing both of these. However, this is demonic, so that the after-state mapped from `w` in U_0 does not compose with anything in the precondition of U_2; we therefore lose the before-state `w` from the domain of $U_0 \fatsemi U_2$, whereas it still exists in the domain of $U_1 \fatsemi U_3$. This is one of the reasons for non-refinement.[5] U_2 S-refines U_3 by weakening its precondition, so in conjunction with the demonic reduction of nondeterminism by U_0, the after-state mapped from `x` in U_0 is composed with the before-state `y` in U_2. This introduces a binding mapping from `x` in $U_0 \fatsemi U_2$ which does not exist in $U_1 \fatsemi U_3$, yet `x` is in the precondition of $U_1 \fatsemi U_3$. This is the second reason for non-refinement. The fact that $(\sqsupseteq_S^+)$ fails for reasons concerning both the precondition and postcondition suggests that neither the SP nor SC refinement theories will support this monotonicity result. Furthermore, it suggests that a sidecondition which is sufficient for proving monotonicity with respect to S-refinement will be needed in both the precondition and postcondition branches of the proof; as we shall see, this is indeed the case.
The counterexample above, and [30, p.39-40],[6] suggest a remedy: if we insist that every after-state in the range of U_1 is mapped onto at least one value in the precondition of U_3, then, not only can strengthening the postcondition (on the left) by U_0 never be demonic (as, in the presence of refinement, the precondition of U_2 is at least as large as the one of U_3), but also weakening the precondition (on the right) by U_2 can never introduce an after-state in $U_0 \fatsemi U_2$ that is connected via an intermediate value that was not in the precondition of U_3.[7] The property is *strong connectivity* and it is defined as follows:

Definition 3.1 (Strong connectivity).

$$Sc\ U_0\ U_1 =_{df} \forall z_0, z_1 \bullet z_0 \star z_1' \in U_0 \Rightarrow Pre\ U_1\ z_1$$

We can prove that schema composition is monotonic with respect to all three refinement theories, providing $Sc\ U_1\ U_3$ holds; but we can do better than that. Although strong connectivity is a very intuitive sidecondition, there is a weaker condition which is also sufficient. This can be motivated by considering further counterexamples. We call it *forking connectivity*. Two specifications comply with this property if, for every nondeterministic before-state (forking point) in the first specification, either *all* the after-states mapped from it connect with some before-state in the *precondition* of the second specification, or *none* of them does.

[5] Note that had the precondition of U_3 not been weakened by U_2, we would have also lost the before-state `x` from the precondition of $U_0 \fatsemi U_2$. This would have induced a chaotic specification $U_0 \fatsemi U_2$.
[6] Grundy proposes a modified *definition* of composition, in which strong connectivity is embedded.
[7] Unless, of course, this is as a result of composing an after-state in U_0 that constitutes a new behaviour (as a consequence of weakening the precondition of U_1) with a before-state outside the precondition of U_3 but which is inside the precondition of U_2. Such a case is not relevant in the present context.

Definition 3.2 (Forking connectivity).

$$Fc\ U_0\ U_1 =_{df} \forall\, z_0, z_1, z_2 \bullet (z_0 \star z_1' \in U_0 \wedge z_0 \star z_2' \in U_0 \wedge Pre\ U_1\ z_1) \Rightarrow Pre\ U_1\ z_2$$

Obvious introduction and elimination rules follow from this.
With this in place, we can now prove the monotonicity result. We shall provide only the proof for S-refinement.

Proposition 3.7. Let U_0, U_1, U_2 and U_3 be operation schemas with the property that:

$$Fc\ U_1\ U_3$$

Then the following rule is derivable:

$$\frac{U_0 \sqsupseteq_s U_1 \quad U_2 \sqsupseteq_s U_3}{U_0 \fatsemi U_2 \sqsupseteq_s U_1 \fatsemi U_3}$$

Proof

$$\dfrac{\dfrac{U_0 \sqsupseteq_s U_1 \quad \dfrac{\overline{Pre\,(U_1 \fatsemi U_3)\,z}\ (1)}{Pre\ U_1\ z}}{Pre\ U_0\ z} \quad \dfrac{\overline{z \star y_0' \in U_0}\ (2) \quad \dfrac{U_2 \sqsupseteq_s U_3 \quad \overset{\alpha_0}{\vdots}\ Pre\ U_3\ y_0}{Pre\ U_2\ y_0}}{Pre\,(U_0 \fatsemi U_2)\,z}}{Pre\,(U_0 \fatsemi U_2)\,z}\ (2) \quad \overset{\delta_1}{\vdots}\ z_0 \star z_1' \in U_1 \fatsemi U_3}{U_0 \fatsemi U_2 \sqsupseteq_s U_1 \fatsemi U_3}\ (1)$$

Where α_0 stands for the following branch:

$$\dfrac{\dfrac{\overline{Pre\,(U_1 \fatsemi U_3)\,z}\ (1)}{Pre\,(U_1 \fatsemi U_3)\,z}\ (1) \quad \dfrac{Fc\ U_1\ U_3 \quad \dfrac{U_0 \sqsupseteq_s U_1 \quad \overline{z \star y_0' \in U_0}\ (2) \quad \dfrac{\overline{Pre\,(U_1 \fatsemi U_3)\,z}\ (1)}{Pre\ U_1\ z}}{z \star y_0' \in U_1} \quad \overline{z \star w_0' \in U_1}\ (3) \quad \overline{Pre\ U_3\ w_0}\ (3)}{Pre\ U_3\ y_0}\ (3)}{Pre\ U_3\ y_0}$$

and δ_1 is:

$$\dfrac{\dfrac{\overline{z_0 \star z_1' \in U_0 \fatsemi U_2}\ (1) \quad \dfrac{\dfrac{U_0 \sqsupseteq_s U_1 \quad \dfrac{\overline{Pre\,(U_1 \fatsemi U_3)\,z_0}\ (1)}{Pre\ U_1\ z_0} \quad \overline{z_0 \star y_1' \in U_0}\ (4)}{z_0 \star y_1' \in U_1} \quad \overset{\alpha_1}{\vdots}\ y_1 \star z_1' \in U_3}{z_0 \star z_1' \in U_1 \fatsemi U_3}\ (4)}{z_0 \star z_1' \in U_1 \fatsemi U_3}$$

Where α_1 stands for the following branch:

$$\dfrac{U_2 \sqsupseteq_s U_3 \quad \dfrac{\overline{Pre\,(U_1 \fatsemi U_3)\,z_0}\ (1) \quad \overset{\beta_1}{\vdots}\ Pre\ U_3\ y_1}{Pre\ U_3\ y_1}\ (5) \quad \overline{y_1 \star z_1' \in U_2}\ (4)}{y_1 \star z_1' \in U_3}$$

Where β_1 stands for the following branch:

$$\dfrac{Fc\ U_1\ U_3 \quad \dfrac{U_0 \sqsupseteq_s U_1 \quad \dfrac{\overline{Pre\,(U_1 \fatsemi U_3)\,z_0}\ (1)}{Pre\ U_1\ z_0} \quad \overline{z_0 \star y_1' \in U_0}\ (4)}{z_0 \star y_1' \in U_1} \quad \overline{z_0 \star w_1' \in U_1}\ (5) \quad \overline{Pre\ U_3\ w_1}\ (5)}{Pre\ U_3\ y_1}$$

□

Finally, it is interesting to note that, since weakening the precondition causes a problem on the right and strengthening

the postcondition causes a problem on the left, it is an immediate consequence that schema composition is *monotonic on the right* with respect to SP-refinement (because the precondition is fixed), and is *monotonic on the left* with respect to SC-refinement (because the postcondition is fixed). Hence, the following rules are derivable:

Proposition 3.8.

$$\frac{U_0 \sqsupseteq_{sp} U_1}{U_2 \fatsemi U_0 \sqsupseteq_{sp} U_2 \fatsemi U_1} \qquad \frac{U_0 \sqsupseteq_{sc} U_1}{U_0 \fatsemi U_2 \sqsupseteq_{sc} U_1 \fatsemi U_2}$$

Proof We provide only the proof for SP-refinement.

$$\dfrac{\dfrac{\dfrac{}{Pre\ (U_2 \fatsemi U_1)\ z}\ (1) \quad \dfrac{\dfrac{}{z \star y' \in U_2}\ (2) \quad \dfrac{U_0 \sqsupseteq_{sp} U_1 \quad \dfrac{}{Pre\ U_1\ y}\ (2)}{Pre\ U_0\ y}}{Pre\ (U_2 \fatsemi U_0)\ z}}{Pre\ (U_2 \fatsemi U_0)\ z} \quad \begin{array}{c}\delta\\ \vdots \\ z_0 \star z_1' \in U_2 \fatsemi U_1\end{array}}{U_2 \fatsemi U_0 \sqsupseteq_{sp} U_2 \fatsemi U_1}\ (1)$$

Where δ is:

$$\dfrac{\dfrac{}{z_0 \star z_1' \in U_2 \fatsemi U_0}\ (1) \quad \dfrac{\dfrac{}{z_0 \star y' \in U_2}\ (3) \quad \dfrac{U_0 \sqsupseteq_{sp} U_1 \quad \dfrac{}{y \star z_1' \in U_0}\ (3)}{y \star z_1' \in U_1}}{z_0 \star z_1' \in U_2 \fatsemi U_1}\ (3)}{z_0 \star z_1' \in U_2 \fatsemi U_1}$$

□

4. Distributivity properties of the *chaotic relational completion* operator

The standard interpretation of refinement for Z in the literature (*e.g.* [65, 13]) is what we have called W$_\bullet$-refinement [21, 20]:

$$U_0 \sqsupseteq_{w_\bullet} U_1 =_{df} \overset{\bullet}{U_0} \subseteq \overset{\bullet}{U_1}$$

where $\overset{\bullet}{U}$ is the lifted-totalisation of U (see appendix A, section A.4 for further detail).

It is important to note that this definition concerns the partial relation interpretation of schema *expressions*. That is, the interpretation of schemas, and of *all* the operations for building modular specifications, are logically *prior* to the theory of refinement.

W$_\bullet$-refinement is, as we have already remarked, equivalent to S-refinement: the theory we used in the previous section. So everything we have established so far also applies to W$_\bullet$-refinement. It is, however, often illuminating to consider matters through distinct though equivalent formulations; this section is devoted to that, mainly through a consideration of lifted-totalisation as an operator in its own right.

One way of illustrating the failure of monotonicity, as it arises in the W$_\bullet$-refinement framework, is to take a look at how the lifted-totalisation interacts directly with the schema operators. For example, if the following full distributivity property held:

$$(U_0 \overset{\bullet}{\wedge} U_1) = \overset{\bullet}{U_0} \wedge \overset{\bullet}{U_1} \qquad \text{✘}$$

then schema conjunction *would* be fully monotonic with respect to refinement. That is, we would have:

$$\frac{U_0 \sqsupseteq_{w_\bullet} U_2 \quad U_1 \sqsupseteq_{w_\bullet} U_3}{U_0 \wedge U_1 \sqsupseteq_{w_\bullet} U_2 \wedge U_3}\ \text{✘}$$

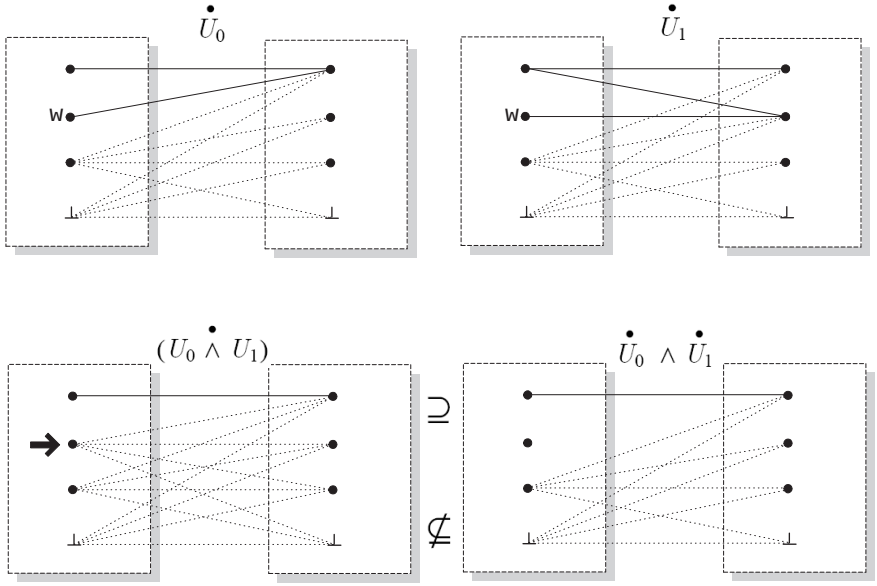

Fig. 4. Lifted-totalisation does not fully distribute over schema conjunction.

with proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{z \in (\dot{U}_0 \wedge \dot{U}_1)}(1)}{z \in \dot{U}_0 \wedge \dot{U}_1}\,\text{✘} \qquad \dfrac{U_0 \sqsupseteq_{w_\bullet} U_2 \qquad z \dot{\in} \dot{U}_0}{z \dot{\in} \dot{U}_2}}{z \in \dot{U}_2 \wedge \dot{U}_3}\,\text{✔} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{z \in (\dot{U}_0 \wedge \dot{U}_1)}(1)}{z \in \dot{U}_0 \wedge \dot{U}_1}\,\text{✘} \qquad \dfrac{U_1 \sqsupseteq_{w_\bullet} U_3 \qquad z \dot{\in} \dot{U}_1}{z \dot{\in} \dot{U}_3}}{z \in (\dot{U}_2 \wedge \dot{U}_3)}}{U_0 \wedge U_1 \sqsupseteq_{w_\bullet} U_2 \wedge U_3}\,(1)$$

Here, the proof annotations indicate the problem: only half of the full distributivity equation holds. Put another way, for full (unconditioned) monotonicity, *we needed the equation at the level of the total-relation semantics, but in Z we have it only at the level of the partial relation semantics* (this is the usual equational logic to be found in the textbooks). The situation is similar for every schema operator. In this section we will analyse in detail the reasons why full distributivity of the lifted-totalisation operator fails with respect to each of the schema calculus operators investigated in section 3. We will introduce sideconditions that are sufficient to attain *full distributivity equations* and then analyse their usefulness and their relationship to the sideconditions introduced in section 3.

4.1. Distributivity for conjunction

The problem with distributing the lifted-totalisation operator over schema conjunction arises when identical before-states of the two component specifications do not agree on their after-states. This leads only to a distributivity inequation and not to full equivalence. The case is illustrated in Fig. 4. The figure illustrates two specifications, U_0 and U_1, which share the before-state `w`, but map it to distinct after-states. Conjoining these two specifications removes `w` from the domain and the completion operator interprets partiality as *chaos*: anything is possible (marked with a right arrow in Fig. 4). This contrasts with the result of conjoining the completions of the specifications, which introduces the partiality at the level of the refinement theory. Here the partiality looks more like *infeasibility*. General infeasibility is often known (*e.g.* in two-predicate frameworks such as the Refinement Calculus and VDM [40, 41]) as *magic*:[8] an

[8] For a complete account of *extreme specifications* see, for example, [48], [30], [60] and [64].

extreme specification whose precondition is *true* and whose postcondition is *false* (it is guaranteed to terminate, yet must establish an impossible outcome). In the figure, the infeasibility is localised: we will refer to this as *local magical behaviour*. [9]

As we can see in Fig. 4, distributing the relational completion operator over schema conjunction may cause local magical behaviour whenever distinct after-states are mapped from the same before-state (`w` in this case) in the two component specifications, prior to the lifted-totalisation. This behaviour results in *partiality*, in a similar fashion to the *two-predicate* based frameworks (including the approach taken in [33] and [37]). Note that Z, a *single-predicate* framework, is capable only of modelling two of the extreme specifications: *chance* and *chaos*, in which the preconditions and postconditions are simultaneously, true and false, respectively. For this reason, we have only a distributivity inequation:

Proposition 4.1. The following rule is derivable:

$$\frac{t_0 \star t_1' \in U_0 \mathbin{\dot{\wedge}} U_1}{t_0 \star t_1' \in (U_0 \mathbin{\dot{\wedge}} U_1)}$$

□

The only way to ensure that distributivity holds in the other direction is by preventing such contentious states in the component specifications: preventing local magical behaviour. This is achieved by insisting that the conjunction of the two specifications will, at least, retain the precondition of *their disjunction*:

Definition 4.1 (Properly conjoined operation schemas).

$$Pc\ U_0\ U_1 =_{df} \forall z \bullet Pre\,(U_0 \vee U_1)\,z \Rightarrow Pre\,(U_0 \wedge U_1)\,z$$

Proposition 4.2. Let U_0 and U_1 be operation schemas with the property that:

$$Pc\ U_0\ U_1$$

Then the following rule is derivable:

$$\frac{t_0 \star t_1' \in (U_0 \mathbin{\dot{\wedge}} U_1)}{t_0 \star t_1' \in U_0 \mathbin{\dot{\wedge}} U_1}$$

□

4.2. Distributivity for disjunction

The lifted-totalisation operator does not fully distribute over schema disjunction because completing component specifications, which have different preconditions, may induce chaotic behaviour in their disjunction, but non-chaotic behaviour when the component specifications are disjoined and then completed. Fig. 5 illustrates this.

[9] Since the appearance of partiality arises here because of over-constraining the compound specification, rather than because of an under-constraint (*e.g.* the partiality in $Predecessor$), there is a good argument for treating it differently (see sections 5.5 and 5.6).

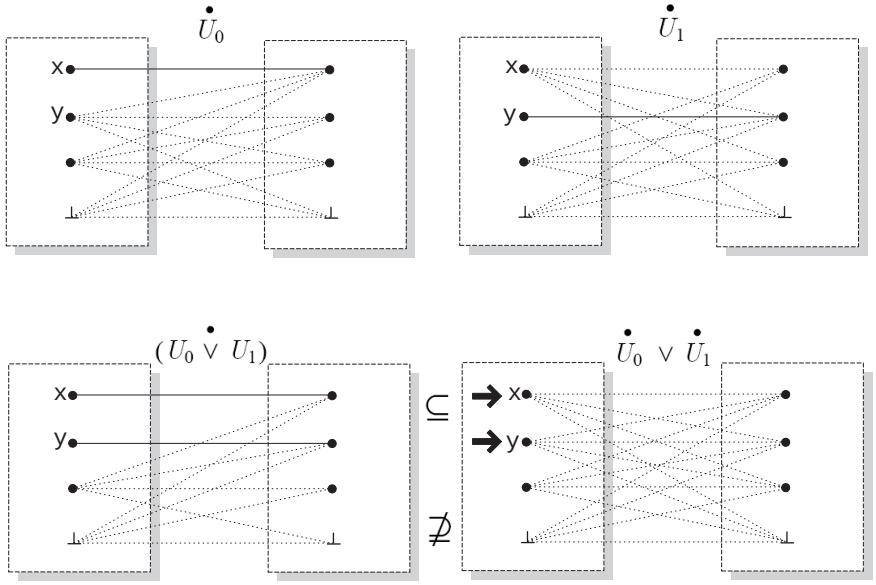

Fig. 5. Lifted-totalisation does not fully distribute over schema disjunction.

The specification U_0 has just one after-state mapped from **x** and the specification U_1 has just one after-state mapped from **y**. Disjoining these partial relations, prior to completion, results in those two after-states (mapped from **x** and **y** in $U_0 \vee U_1$) and chaotic behaviour everywhere else. However, applying the lifted-totalisation to U_0 and U_1 (individually), gives rise to chaotic behaviour mapped from **y** in $\overset{\bullet}{U_0}$ and similarly for **x** in $\overset{\bullet}{U_1}$. Thus, $\overset{\bullet}{U_0} \vee \overset{\bullet}{U_1}$ is chaotic from these two before-states (marked with right arrows); hence, we have an inequation rather than a full equivalence.

Proposition 4.3. The following rule is derivable:

$$\frac{t_0 \star t_1' \in (U_0 \overset{\bullet}{\vee} U_1)}{t_0 \star t_1' \in \overset{\bullet}{U_0} \vee \overset{\bullet}{U_1}}$$

□

We observed, in Fig. 5, that full distributivity fails, because of distinctions in the preconditions of the specifications U_0 and U_1. Therefore, insisting that the component specifications have identical preconditions guarantees full distributivity:

Definition 4.2 (Stable preconditions).

$$Sp\ U_0\ U_1 =_{df} \forall z \bullet Pre\ U_0\ z \Leftrightarrow Pre\ U_1\ z$$

Proposition 4.4. Let U_0 and U_1 be operation schemas with the property that:

$$Sp\ U_0\ U_1$$

Then the following rule is derivable:

$$\frac{t_0 \star t_1' \in \overset{\bullet}{U_0} \vee \overset{\bullet}{U_1}}{t_0 \star t_1' \in (U_0 \overset{\bullet}{\vee} U_1)}$$

□

4.3. Distributivity for existential quantification

Fully distributing the relational completion operator over schema existential hiding fails. This is because hiding observations after applying lifted-totalisation can introduce chaotic behaviour that will not always arise when hiding observations before lifted-totalisation. This is shown in Fig. 6.

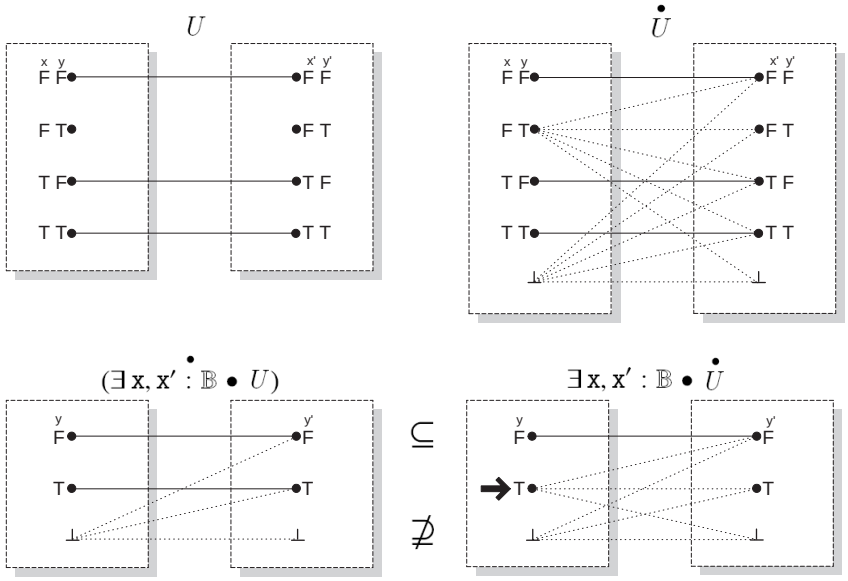

Fig. 6. Lifted-totalisation does not fully distribute over schema existential quantification.

We present a specification U whose alphabet comprises the boolean observations $\mathtt{x}$, $\mathtt{x}'$, $\mathtt{y}$ and $\mathtt{y}'$. Hiding the observations $\mathtt{x}$ and $\mathtt{x}'$ yields a *total* specification $\exists\, \mathtt{x}, \mathtt{x}' : \mathbb{B} \bullet U$. The only effect of lifted-totalisation on this will be the mapping of $\bot$ onto all after-states. Yet, hiding the same observations after lifted-totalisation introduces chaotic behaviour from the before-state $\mathtt{T}$ (marked with a right arrow) in the specification $\exists\, \mathtt{x}, \mathtt{x}' : \mathbb{B} \bullet \dot{U}$. This is a consequence of mapping the before-state $\mathtt{FT}$ (which is outside the precondition of U) onto all the after-states in $\dot{U}$ (including $\bot$). As a result, hiding the observations $\mathtt{x}$ and $\mathtt{x}'$ in $\dot{U}$ leaves the remainder of this state sanctioning every possible outcome. For this reason, we have only a distributivity inequation:

Proposition 4.5. The following rule is derivable:

$$\frac{t \in (\exists\, \mathtt{z} : \dot{T^{\mathtt{z}}} \bullet U)}{t \in \exists\, \mathtt{z} : T^{\mathtt{z}} \bullet \dot{U}}$$

□

Since existential quantification is a generalisation of disjunction, it is not surprising that the failure of the converse inequation is reminiscent of the case for schema disjunction (section 4.2), although here we have one specification rather than two: the difference described by Fig. 6 arises because distinct before-states, which involve the same hidden observation in the specification, have different *precondition status*.[10]
Naturally, the remedy is very similar to "stable preconditions" (definition 4.2), though here we have only a single specification: we need to ensure that any before-state in the precondition of $\exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U$ is equivalently in the precondition of U. One direction is merely ($Pre^{+}_{\exists}$) (see appendix A, section A.3.3); thus all we need is the following property:

Definition 4.3 (Weak binding). $Wb\ U =_{df} \forall\, x \bullet Pre\,(\exists\, \mathtt{z} : T^{\mathtt{z}} \bullet U)\, x \Rightarrow Pre\ U\ x$

[10] One is in the precondition and one is outside the precondition of the specification.

Proposition 4.6. Let U be operation schema with the property that:

$$Wb\ U$$

Then the following rule is derivable:

$$\frac{t \in \exists\, \mathtt{z} : T^{\mathtt{z}} \bullet \dot{U}}{t \in (\exists\, \mathtt{z} : \dot{T}^{\mathtt{z}} \bullet U)}$$

□

4.4. Distributivity for composition

The lifted-totalisation operator distributes over schema composition (but not conversely) because composing a non-deterministic specification (on the left) with a partial specification (on the right) may give rise to local chaos in the composition of their (individual) completions, but which might not arise in the completion of their composition. This is illustrated in Fig. 7.

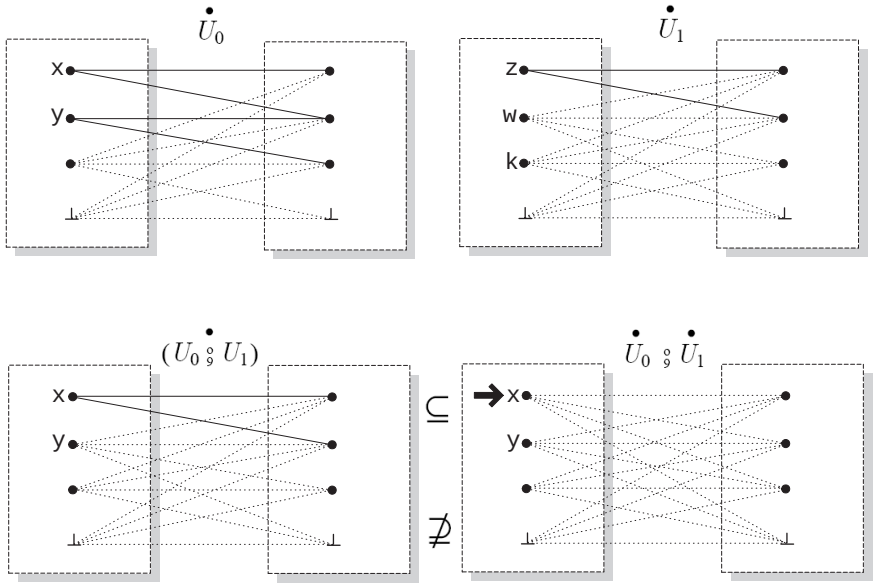

Fig. 7. Lifted-totalisation does not fully distribute over schema composition.

An observation, such as $\mathtt{w}$, outside the precondition of U_1 plays no part in linking before-states of U_0 and after-states of U_1 in the composition $U_0 \fatsemi U_1$ (nor its lifted-totalisation). Thus $\mathtt{x}$, in $U_0 \fatsemi U_1$, is associated with the two after-states, mapped from $\mathtt{z}$ in U_1. However, applying the relational completion operator to U_0 and U_1 separately, results in chaotic behaviour mapped from $\mathtt{w}$ in $\dot{U}_1$ and consequently chaotic behaviour (marked with a right arrow) from $\mathtt{x}$ in $\dot{U}_0 \dot{\fatsemi} \dot{U}_1$. Therefore, we have only the following inequation. We provide the proofs in this case.

Proposition 4.7. The following rule is derivable:

$$\frac{t_0 \star t_1' \in (U_0 \dot{\fatsemi} U_1)}{t_0 \star t_1' \in \dot{U}_0 \dot{\fatsemi} \dot{U}_1}$$

Proof

$$\dfrac{\dfrac{t_0 \star t_1' \in (U_0 \dot{\fatsemi} U_1) \qquad \overset{\delta_0}{\vdots}\ t_0 \star t_1' \in \dot{U}_0 \dot{\fatsemi} \dot{U}_1}{t_0 \star t_1' \in \dot{U}_0 \dot{\fatsemi} \dot{U}_1} \qquad \dfrac{\dfrac{\dfrac{}{t_0 \star t_1' \in U_0 \fatsemi U_1}\,(1)}{t_0 \star t_1' \in \dot{U}_0 \dot{\fatsemi} \dot{U}_1}\,(5) \qquad \dfrac{\dfrac{\dfrac{}{t_0 \star y' \in U_0}\,(5)}{t_0 \star y' \in \dot{U}_0}\,(L.\,A.3(i)) \qquad \dfrac{\dfrac{}{y \star t_1' \in U_1}\,(5)}{y \star t_1' \in \dot{U}_1}\,(L.\,A.3(i))}{t_0 \star t_1' \in \dot{U}_0 \dot{\fatsemi} \dot{U}_1}}{t_0 \star t_1' \in \dot{U}_0 \dot{\fatsemi} \dot{U}_1}\,\heartsuit(1)}$$

Where δ_0 stands for the following branch:

$$\dfrac{\dfrac{\dfrac{}{\neg\ Pre\,(U_0 \fatsemi U_1)\ t_0}\ (1)}{\neg\ Pre\ U_0\ t_0 \lor (\forall\, z \bullet t_0 \star z' \in U_0 \Rightarrow \neg\ Pre\ U_1\ z)} \quad \overset{\delta_1}{\vdots}\ t_0 \star t_1' \in U_0 \fatsemi U_1 \quad \overset{\delta_2}{\vdots}\ t_0 \star t_1' \in U_0 \fatsemi U_1}{t_0 \star t_1' \in U_0 \fatsemi U_1}\ (2)$$

Where δ_1 is:

$$\dfrac{\dfrac{}{\neg\ Pre\ U_0\ t_0}\ (2) \quad \dfrac{\dfrac{t_0 \star t_1' \in (U_0 \fatsemi U_1)}{t_0 \star t_1' \in T_{0_\perp}^{in} \star T_{1_\perp}^{out'}}}{t_0 \in T_{0_\perp}^{in}}\ (L.\ A.3(iv)) \qquad \dfrac{\dfrac{t_0 \star t_1' \in (U_0 \fatsemi U_1)}{t_0 \star t_1' \in T_{0_\perp}^{in} \star T_{1_\perp}^{out'}}}{t_1' \in T_{1_\perp}^{out'}}\ (L.\ A.3(iii))}{\dfrac{t_0 \star \perp' \in U_0 \qquad \perp \star t_1' \in U_1}{t_0 \star t_1' \in U_0 \fatsemi U_1}}$$

and δ_2 is:

$$\dfrac{\dfrac{}{Pre\ U_0\ t_0 \lor \neg\ Pre\ U_0\ t_0}\ (LEM) \quad \dfrac{\dfrac{}{Pre\ U_0\ t_0}\ (3) \quad \dfrac{\dfrac{\dfrac{}{t_0 \star w' \in U_0}\ (4)}{t_0 \star w' \in U_0}\ (L.\ A.3(i)) \quad \overset{\beta_0}{\vdots}\ w \star t_1' \in U_1}{t_0 \star t_1' \in U_0 \fatsemi U_1}\ (4) \quad \overset{\beta_1}{\vdots}\ t_0 \star t_1' \in U_0 \fatsemi U_1}{t_0 \star t_1' \in U_0 \fatsemi U_1}\ (3)}{t_0 \star t_1' \in U_0 \fatsemi U_1}$$

Where β_0 is:

$$\dfrac{\dfrac{\dfrac{}{\forall\, z \bullet t_0 \star z' \in U_0 \Rightarrow \neg\ Pre\ U_1\ z}\ (2)}{t_0 \star w' \in U_0 \Rightarrow \neg\ Pre\ U_1\ w} \quad \dfrac{}{t_0 \star w' \in U_0}\ (4)}{\neg\ Pre\ U_1\ w} \quad \dfrac{\dfrac{\dfrac{}{t_0 \star w' \in U_0}\ (4)}{t_0 \star w' \in T_0^{in} \star T_1^{in'}}}{\dfrac{w \in T_1^{in}}{w \in T_{1_\perp}^{in}}} \quad \dfrac{t_0 \star t_1' \in (U_0 \fatsemi U_1)}{\dfrac{t_0 \star t_1' \in T_{0_\perp}^{in} \star T_{1_\perp}^{out'}}{t_1' \in T_{1_\perp}^{out'}}}}{w \star t_1' \in U_1}\ (L.\ A.3(v))$$

and β_1 is identical to δ_1 modulo the substitution of the label (*3*) for the label (*2*). □

There are two observations we can make of the proof. First, note that the proof step labelled ♡ denotes an application of $(\bullet^-)$ but where the definition A.10 is expressed using disjunction (in the obvious way) in place of implication, leading to a single disjunctive elimination rule. This rule is also used in the proofs for propositions 4.3 and 4.5. Second, the proof depends on use of the *law of excluded middle*. We suspect that this result is strictly classical, and there appear to be many other similar examples in refinement theory (*e.g.* [15, 14, 16]), so abandoning the *constructive approach* for Z taken in, for example, [34, 35] and [46] may be inevitable.

Fig. 7 demonstrates that distributivity fails in the other direction precisely because the two after-states mapped from $\mathtt{x}$ in U_0 coincide with two before-states in U_1 with different precondition status. We can see that the forking point $\mathtt{y}$ in U_0 does not constitute a problem, since $\mathtt{w}$ and $\mathtt{k}$ are both outside the precondition of U_1; thus $\mathtt{y}$ is associated with chaotic behaviour in both $(U_0 \fatsemi U_1)$ and $U_0 \fatsemi U_1$. Furthermore, had we had $\mathtt{w}$ in the precondition of U_1, we would have obtained the same (non-chaotic) behaviour associated with $\mathtt{x}$ in both cases. This suggests that a sidecondition guaranteeing full distributivity would insist on associating all the after-states, mapped from a certain *nondeterministic before-state* in U_0, with some before-states in U_1 – all of which have the same precondition status. This is, indeed, the *forking connectivity* property used to ensure monotonicity of schema composition with respect to S-refinement (section 3.4).

Proposition 4.8. Let U_0 and U_1 be operation schemas, with the property that:

$$Fc\ U_0\ U_1$$

Then the following rule is derivable:

$$\frac{t_0 \star t_1' \in \dot{U}_0 \mathbin{\dot{\circ}} \dot{U}_1}{t_0 \star t_1' \in (U_0 \mathbin{\dot{\circ}} U_1)}$$

Proof

$$\dfrac{\dfrac{\dfrac{\beta}{t_0 \star y' \in U_0} \quad \dfrac{\dfrac{}{y \star t_1' \in \dot{U}_1}\,(1) \quad \overset{\delta}{\vdots}\ Pre\ U_1\ y}{y \star t_1' \in U_1}}{t_0 \star t_1' \in U_0 \mathbin{\dot{\circ}} U_1} \quad \dfrac{t_0 \star t_1' \in \dot{U}_0 \mathbin{\dot{\circ}} \dot{U}_1}{t_0 \star t_1' \in T_{0_\perp}^{in} \star T_{1_\perp}^{out'}}}{t_0 \star t_1' \in \dot{U}_0 \mathbin{\dot{\circ}} \dot{U}_1 \qquad t_0 \star t_1' \in (U_0 \mathbin{\dot{\circ}} U_1)}\,(2)}{t_0 \star t_1' \in (U_0 \mathbin{\dot{\circ}} U_1)}\,(1)$$

Where δ stands for the following branch:

$$\dfrac{\dfrac{}{Pre\,(U_0 \mathbin{\dot{\circ}} U_1)\ t_0}\,(2) \quad \dfrac{Fc\ U_0\ U_1 \quad \overset{\beta}{\vdots}\ t_0 \star y' \in U_0 \quad \dfrac{}{Pre\ U_1\ w}\,(3)}{Pre\ U_1\ y}\,(3) \quad \dfrac{}{t_0 \star w' \in U_0}\,(3)}{Pre\ U_1\ y}\,(3)$$

and β stands for the following branch:

$$\dfrac{\dfrac{}{t_0 \star y' \in \dot{U}_0}\,(1) \quad \dfrac{\dfrac{}{Pre\,(U_0 \mathbin{\dot{\circ}} U_1)\ t_0}\,(2)}{Pre\ U_0\ t_0}\,(L.\ A.2)}{t_0 \star y' \in U_0}$$

□

4.5. Discussion

The sideconditions we have isolated, either for ensuring full distributivity or for establishing monotonicity directly, are not similar to the syntactic sideconditions routinely associated with logical rules, such as ∃-elimination. Syntactic sideconditions are decidable, so it can always be determined when a rule applies. The sideconditions we have formulated are proof-theoretically more complex and, in fact, only semi-decidable. From a practical point of view this is not very satisfactory. Moreover, most make mention of the *concrete* specification in addition to the abstract specification. This is unfortunate because it reduces the practical use of the refinement rules when used *calculationally*: to *construct* the concrete specification. The exception to this are the two connectivity principles used in the case of composition. And one could in fact avoid mention of the concrete specification, in the case of disjunction, by omitting one of the antecedent propositions: thereby obtaining a condition which is purely abstract, but which is, of course, somewhat less applicable. It could be argued that these sideconditions are reasonable only if they refer *exclusively* to either the abstract or the concrete specifications [57]. In this way the true spirit of *abstraction* (in which the internal structure of the abstract specification is not to be disclosed) is upheld. Overall the lack of monotonicity is a distinct drawback which such proof-theoretic sideconditions do not address satisfactorily from a practical perspective.
Our analysis has of course concentrated on only two (equivalent) notions of refinement: S-refinement and W$_\bullet$-refinement. Possibly, there are other formulations of refinement which would be better behaved. And there are, in fact, several alternative approaches:

- *Weakest precondition refinement* – it is possible to reinterpret the partial relations in terms of a weakest precondition semantics and to characterise refinement in the standard way in that regime;

- *Sets of implementation* – in the spirit of constructive theories of program development, *e.g.* Martin-Löf Type Theory [45] (though in the setting of classical logic) it is possible to reinterpret specifications as sets of permissible implementations. Refinement in this case is simply set inclusion;
- *Strict-lifted-totalisation* – it is possible to modify the lifted-totalisation so that the lifting is strict (abortive) rather then non-strict (chaotic);
- *Non-lifted-totalisation* – it is possible to totalise the partial relations without lifting if one is prepared to exclude fully chaotic behaviour from the notion of precondition.

We demonstrated in [21] that all of these theories of refinement are equivalent to the standard lifted-totalised account. As a consequence, all suffer from the same weaknesses in terms of their (lack of) monotonicity properties. Naturally, one could ask: are there still others which have yet to be discovered? In addressing this question we would need to find some *principles* which distinguish a relation worthy of the name "refinement" from any arbitrary binary relation on specifications. After all, the schema operators are all fully monotonic with respect to *equality*, but equality is evidently *not* a notion of refinement. In capturing the general principles one will be led to the properties described by S-refinement or SP-refinement. A notion of refinement ought to be at least *sound* with respect to one or other of those. Our analysis already demonstrates the limits of modular reasoning with respect to these notions.
The analysis of distributivity in this section does include an interesting clue which motivates our final section. This was the observation that we have an equational logic at the level of the underlying *partial* relations, but not at the level of the *total* relations involved in refinement. This suggests an alternative approach. Instead of developing a schema calculus at the level of partial relations and only then introducing total relational refinement, we could introduce that calculus afterwards. This would naturally lead to a fully monotonic schema calculus, because refinement would then simply become the subset relation on the modified relational model. Of course it would lead to a distinct schema calculus with possibly quite different properties. These would also need to be investigated. The latter investigation is beyond the scope of this paper; but the results of the following section suggest that it would be interesting to pursue.

5. A fully monotonic schema calculus

Z, as we have seen, takes a *layered* approach to refinement: the underlying semantics of schema expressions is partial relational, but refinement is based on a subsequent interpretation. This may take many guises (including the standard, lifted-totalisation model) but all lead to equivalent theories for which the schema operators are not monotonic. Z does however have an equational logic (section 1.2).
This leaves us with an interesting question: *Which is more important: the equational logic or monotonicity of the schema operations?* – evidently we cannot have both simultaneously. Historically, the clear answer is: *the equational logic*, though this may be because, until very recently, the lack of a mathematical analysis precluded addressing what was otherwise an unasked question. Obviously, there is no definitive answer: the matter would have to be settled by features of the application context. However, since one answer to the question has been thoroughly investigated (that is Z, as we understand it), it will be interesting to explore the consequences of the alternative answer: if we abandon the equational logic we may be able to rehabilitate monotonicity.
We introduce, in this section, one possible approach for attaining a fully monotonic schema calculus. Motivated by the distributivity results in the preceding section, we replace the usual partial relation semantics with the lifted-totalised interpretation, taking it *directly* as the semantics for atomic schemas and then introducing the meaning of compound operation schema expressions by recursion over their structure, using the standard interpretation of the schema operators. In this way, refinement reduces to the subset relation on the new semantics, and the schema calculus becomes fully monotonic.

5.1. Logic and semantics

The new semantics (written $[\![\_]\!]_1$) is defined as follows:

Definition 5.1. Let U be an atomic operation schema, then:

$$[\![U]\!]_1 =_{df} \dot{U}$$

Extending this to all schema expressions is compositional: as given by definition A.8 for the standard interpretation.

5.2. T$_\bullet$-refinement

Refinement based on the new *totalised* semantics (hence "T"$_\bullet$-refinement) is written $U_0 \sqsupseteq_{t_\bullet} U_1$ and is simply defined as a subset relation on the new semantics. The term "totalised" here refers to the approach taken for the interpretation of atomic schemas: under-specification is modelled in this theory of refinement as chaos. Note that schema expressions in general still denote partial relations because we use the standard interpretation of conjunction and composition. Over-specification is modelled in this theory of refinement as magic (see section 4.1 for our earlier discussion of this).

Definition 5.2.

$$\left[\!\left[U_0 \sqsupseteq_{t_\bullet} U_1 \right]\!\right]_1 =_{df} [\![U_0]\!]_1 \subseteq [\![U_1]\!]_1$$

The following introduction and elimination rules are derivable for T$_\bullet$-refinement:

Proposition 5.1. Let z be a fresh variable.

$$\frac{z \in U_0 \vdash z \in U_1}{U_0 \sqsupseteq_{t_\bullet} U_1}\ (\sqsupseteq^+_{t_\bullet}) \qquad \frac{U_0 \sqsupseteq_{t_\bullet} U_1 \quad t \in U_0}{t \in U_1}\ (\sqsupseteq^-_{t_\bullet})$$

□

5.2.1. Refinement logic

In this model we lose the usual equational logic for the operation schema calculus, trading this for refinement inequations. Put another way: the various semi-distributivity results presented earlier (section 4) become, under this model, an *inequational logic*.

Proposition 5.2 (Conjunction inequation).

$$[D_0 \mid P_0] \wedge [D_1 \mid P_1] \sqsupseteq_{t_\bullet} [D_0;\ D_1 \mid P_0 \wedge P_1]$$

Proof Follows by proposition 4.1 and [36] – proposition 5.5. □

Proposition 5.3 (Disjunction inequation).

$$[D_0;\ D_1 \mid P_0 \vee P_1] \sqsupseteq_{t_\bullet} [D_0 \mid P_0] \vee [D_1 \mid P_1]$$

Proof Follows by proposition 4.3 and [36] – proposition 4.10. □

Proposition 5.4 (Existential quantification inequation).

$$[D \mid \exists\, u : T^{\mathtt{z}} \bullet P[\mathtt{z}/u]] \sqsupseteq_{t_\bullet} \exists\, \mathtt{z} : T^{\mathtt{z}} \bullet [\mathtt{z} : T^{\mathtt{z}};\ D \mid P]$$

Proof Follows by proposition 4.5 and [36] – proposition 4.11. □

Proposition 5.5 (Composition inequation). Let $T^{in} = [\mathtt{x} : T]$ and $T^{out'} = [\mathtt{x}' : T]$.

$$[\mathtt{x}, \mathtt{x}' : T \mid \exists\, v^{T^{in}} \bullet P_0[\mathtt{x}'/v'.\mathtt{x}'] \wedge P_1[\mathtt{x}/v.\mathtt{x}]] \sqsupseteq_{t_\bullet} [\mathtt{x}, \mathtt{x}' : T \mid P_0] \fatsemi [\mathtt{x}, \mathtt{x}' : T \mid P_1]$$

Proof Follows by propositions 4.7 and A.2. □

5.3. Monotonicity

All the schema operators defined for the new semantics in section 5.1 are monotonic with respect to T$_\bullet$-refinement. The proofs for the following monotonicity properties are trivial: they all follow because the semantics fully distributes over the standard relational operations.

Proposition 5.6. All four schema operators are monotonic with respect to T$_\bullet$-refinement.

$$\frac{U_0 \sqsupseteq_{t_\bullet} U_1}{U_0 \wedge U_2 \sqsupseteq_{t_\bullet} U_1 \wedge U_2}\ (i) \qquad \frac{U_0 \sqsupseteq_{t_\bullet} U_1}{U_0 \vee U_2 \sqsupseteq_{t_\bullet} U_1 \vee U_2}\ (ii)$$

$$\frac{U_0 \sqsupseteq_{t_\bullet} U_1}{\exists \mathsf{z} : T^{\mathsf{z}} \bullet U_0 \sqsupseteq_{t_\bullet} \exists \mathsf{z} : T^{\mathsf{z}} \bullet U_1} \ (iii) \qquad \frac{U_0 \sqsupseteq_{t_\bullet} U_1 \quad U_2 \sqsupseteq_{t_\bullet} U_3}{U_0 \fatsemi U_2 \sqsupseteq_{t_\bullet} U_1 \fatsemi U_3} \ (iv)$$

□

5.4. The relationship between W•-refinement and T•-refinement

What is the relationship between T•-refinement and the standard characterisation of refinement in Z, (what we have denoted as) W•-refinement?
The answer follows from the investigation conducted in section 4: since T•-refinement relies on distributing the semantics over the *standard schema algebra*, W•-refinement would be equivalent to T•-refinement only for those schema expressions satisfying the conditions for full distributivity of the lifted-totalisation over the standard operations. We will now formalise this relationship, highlighting the specific use of the sideconditions (section 4) in the process.
In order to manage both interpretations of schemas simultaneously, we need to distinguish between membership in the two schema calculi. To this end we will use $\in_0$ in the standard theory and $\in_1$ in the new semantics. More exactly, we have the standard model (see appendix A):

$$[\![t \in_0 U]\!]_0 =_{df} [\![t]\!]_0 \in [\![U]\!]_0$$

and, for the new interpretation:

$$[\![t \in_1 U]\!]_1 =_{df} [\![t]\!]_1 \in [\![U]\!]_1$$

Proposition 5.7. Let U range over only those schema expressions which satisfy the following:

(*i*) Schema disjunction expressions satisfy ***stable preconditions*** (definition 4.2);
(*ii*) Existentially quantified schema expressions satisfy ***weak binding*** (definition 4.3);
(*iii*) Schema composition expressions satisfy ***forking connectivity*** (definition 3.2).

Then the following rule is derivable:

$$\frac{t \in_1 U}{t \in_0 \overset{\bullet}{U}}$$

Proof By induction over the structure of U using propositions 4.1, 4.4, 4.6 and 4.8. □

Proposition 5.8. Let U range over only those schema expressions for which schema conjunction subexpressions are ***properly conjoined*** (definition 4.1). Then the following rule is derivable:

$$\frac{t \in_0 \overset{\bullet}{U}}{t \in_1 U}$$

Proof By induction over the structure of U using propositions 4.2, 4.3, 4.5 and 4.7. □

Proving that W•-refinement implies T•-refinement (under certain circumstances) is now straightforward:

Theorem 5.1. Let U_0 range over that subset of schema expressions satisfying the sideconditions stated in proposition 5.7 and let U_1 range over that subset of schema expressions satisfying the sidecondition stated in proposition 5.8. Then the following is derivable:

$$\frac{U_0 \sqsupseteq_{w_\bullet} U_1}{U_0 \sqsupseteq_{t_\bullet} U_1}$$

□

Likewise:

Theorem 5.2. Let U_0 range over a subset of schema expressions, which satisfy the sidecondition stated in proposition 5.8 and Let U_1 range over a subset of schema expressions satisfying the sideconditions stated in proposition 5.7. Then the following is derivable:

$$\frac{U_0 \sqsupseteq_{t_\bullet} U_1}{U_0 \sqsupseteq_{w_\bullet} U_1}$$

□

Together, theorems 5.1 and 5.2 establish that the theories of $T_\bullet$-refinement and $W_\bullet$-refinement are equivalent for schema expressions satisfying full distributivity of the relational completion operator with respect to the various schema operations.

5.5. Discussion

Apart from the lack of full distributivity with respect to the schema algebra (section 4), the standard interpretation has an additional flaw: recall that, as mentioned in section 4.1, the standard theory of refinement for Z interprets *all* partiality chaotically, as under-specification.
Consider the following example:

$$U_0 \mathrel{\widehat{=}} [\, \mathtt{x}, \mathtt{x}' : \mathbb{N} \mid \mathtt{x}' = 1 \,] \qquad U_1 \mathrel{\widehat{=}} [\, \mathtt{x}, \mathtt{x}' : \mathbb{N} \mid \mathtt{x}' = 2 \,] \qquad U_2 \mathrel{\widehat{=}} [\, \mathtt{x}, \mathtt{x}' : \mathbb{N} \mid \mathtt{x}' = 8 \,]$$

The conjunction of U_0 and U_1, given the equational logic, is equivalent to:

$$U_0 \wedge U_1 = [\, \mathtt{x}, \mathtt{x}' : \mathbb{N} \mid \mathtt{x}' = 1 \wedge \mathtt{x}' = 2 \,]$$

This model is nowhere defined and, given that the standard notion of refinement in Z is merely a subset relation on the totalisations of the (atomic) specifications, this is $Chaos$ which may be refined by any specification [30]. Therefore we have:

$$U_2 \sqsupseteq_{w_\bullet} U_0 \wedge U_1$$

Yet, is this a reasonable refinement?
Imagine a safety-critical specification, for which two independent assessments of certain inputs must combine (conjunctively) to create safe conditions on certain outputs. If circumstances arise in which those independent assessments can conflict, then it will be possible to make arbitrary refinement choices in conditioning the outputs at that point of conflict. When the system is sufficiently complex as to require machine support for its refinement, a *mathematically correct refinement* (indeed a machine-checked system development) might result in unnoticed absurdities and, of course, the possibility that the final system is anything but safe. It should be stressed that this is a flaw in the *conceptualisation* and not in the mathematical account (which is of course perfectly sound): mathematical correctness is, in itself, an insufficient guarantee.
Note that, interpreted in the new semantics, the inequation fails; indeed, since refinement is simply subset on the model, partiality can never be removed by refinement. In the new theory of refinement, partiality is treated chaotically in those cases which arise from under-specification (in atomic schemas) and treated magically in cases where it arises from over-specification (in some instances of conjunction or composition). In particular, since it is not possible to *implement* magic [47, 48, 30], nothing, including U_2, can refine $U_0 \wedge U_1$.

5.6. Some alternative approaches

An alternative approach, suggested by Dunne [25], is based on the *abortive-lifted-totalisation*, which we denote as $\overset{\Box}{U}$ (see appendix A, definition A.12 and [20]); this is effectively $T_\Box$-refinement, that is, refinement based on the general approach described above, but where definition 5.1 is replaced by:

$$[\![U]\!]_2 =_{df} \overset{\Box}{U}$$

The abortive-lifted-totalisation operator is characterised by the same distributivity properties as its chaotic counterpart (section 4) and hence $T_\Box$-refinement provides a similar *inequational logic* to the one for $T_\bullet$-refinement (section 5.2.1) only with an underlying *abortive* semantics. Of course $T_\Box$-refinement precludes the weakening of preconditions, which is, for many applications, a weakness and is not one shared by the model we introduced earlier.

In light of the analysis provided in [21], [15] and [17, 14], in which the strict-lifted-totalisation, denoted $\overset{\ominus}{U}$ (appendix A, definition A.11), has the same effect in model-theoretic refinement as its non-strict counterpart, one may be inclined to suppose that $T_\ominus$-refinement[11] is equivalent to $T_\bullet$-refinement. In fact the strict-lifted-totalisation does not share

[11] $T_\ominus$-refinement is based on the model where definition 5.1 is replaced by: $[\![U]\!]_3 =_{df} \overset{\ominus}{U}$.

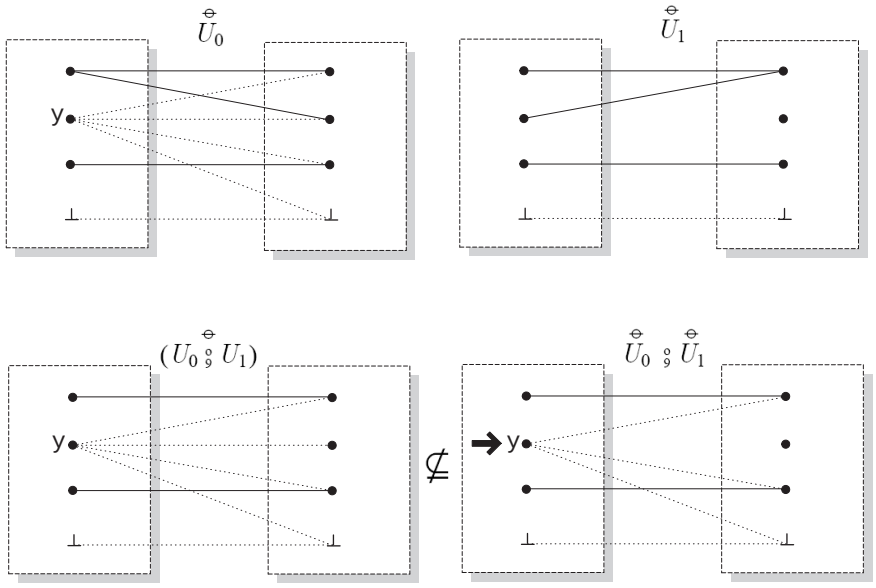

Fig. 8. Strict-lifted-totalisation does not distribute over schema composition.

distributivity properties with the other two models and, as a result, $T_{\ominus}$-refinement and $T_{\bullet}$-refinement are *not equivalent*. This difference stems from the fact that, unlike $\dot{U}$ and $\overset{\Box}{U}$, $\perp$ in $\hat{\ominus} U$ maps only to the after-state $\perp$, whereas any other before-state outside the precondition maps chaotically. Note that in the analysis of section 4, the only result requiring an explicit use of lemma A.3(iii), indicating the *non-strictness* of $\perp$, is proposition 4.7; hence it is bound to fail in the strict model. In fact, none of the other results crucially involve $\perp$, so they all trivially hold in the strict model.
The reason for losing the unconditioned distributivity inequation for composition in this model is demonstrated by Fig. 8. We introduce a partial specification U_0 (the before-state y is outside its precondition) and a total specification U_1. The partiality of U_0 induces partiality for their composition, thus y is locally chaotic in $(U_0 \hat{\ominus}_{\S} U_1)$ (as well as in its non-strict counterpart). Therefore, in order for the distributivity inequation to hold, it is essential that y is locally chaotic in $\hat{\ominus} U_0 \ \hat{\ominus}_{\S} \ \hat{\ominus} U_1$. Attaining this when distributing the non-strict-lifted-totalisation is trivial, because y is mapped onto $\perp$ (as well as to everything else) in $\dot{U}_0$ and $\perp$ is mapped onto all the after-states in $\dot{U}_1$. Hence, in any case, y will be locally chaotic in $\dot{U}_0 \ \dot{\S} \ \dot{U}_1$. In contrast, local chaos might not be produced when distributing the strict model over composition (marked with a right arrow in Fig. 8). This occurs because U_0 is partial and U_1 is total *but not surjective*. Hence, the remedy is the following sidecondition:

Definition 5.3 (Totality dependency).

$$Td\ U_0\ U_1 =_{df} total(U_1) \Rightarrow total(U_0) \vee surjective(U_1)$$

Where $total$ and $surjective$ are defined as usual.

In which case:

$$(U_0 \hat{\ominus}_{\S} U_1) \subseteq \hat{\ominus} U_0 \ \hat{\ominus}_{\S} \ \hat{\ominus} U_1$$

In fact, the distributivity inequations for composition are *counter-dependent* in the strict model: when one inequation fails the other necessarily succeeds. This is a direct consequence of the property that: $total(U_1) \Rightarrow Fc\ U_0\ U_1$.
However, the above sidecondition is very strong and the lack of an unconditioned inequation for composition in the strict model has an important consequence: we *lose the refinement inequation for composition* (proposition 5.5) in the $T_{\ominus}$-refinement model.[12] It would be very interesting to explore the pragmatic consequences of that.

[12] In line with the results in [21, 15, 17, 14], this would be the only reason to prefer Woodcock's non-strict model to its strict counterpart.

6. Conclusions and future work

This paper provides a thorough investigation of monotonicity properties for the schema calculus of Z, with respect to a variety of notions of operation refinement.

In the first and second parts we carefully examined four schema calculus operators, looking at monotonicity properties directly and at distribution properties with respect to the lifted-totalisation operation which lies at the heart of the standard account of refinement in Z. In addition to demonstrating the lack of monotonicity properties, we provided a number of sideconditions which are sufficient for guaranteeing monotonicity – although these conditions are proof-theoretic, rather than syntactic, in character and, therefore, arguably of limited use.

The third part of the paper introduced a novel semantics for schemas which is trivially monotonic. In exchange for monotonicity we must give up the usual equational logic, trading it for an *inequational* logic of refinement. In many ways this is unexceptional and unobjectionable: most formal methods take refinement as the *basic* relation; Z is unusual in insisting on equality and, as we have seen, it is this priority for equality which is responsible for the lack of useful modular properties for refinement in the standard account. Possibly the reason for this is essentially historical: Z was one of the first methods to be established and it was some time before refinement entered the scene. Moreover, it appears that it was originally conceived as essentially a simple *notation* for presenting mathematical descriptions, with a distinct emphasis on syntax rather than logic or semantics. Similarly, the schema operations were essentially a notational short-hand: what we now think of as equations (which *hold* for the schema operations) were originally simply their *definitions*.

Naturally, the loss of the usual equational logic means that the schema operations, in the new semantics, *have entirely different properties* from those they have in the standard account. Quite apart from the theoretical questions, some of which we have addressed here, are equally important *pragmatic* questions. How does one *use* an alternative schema calculus to construct specifications? What is a useful family of such operations? Does the theory of refinement make *practical* sense? This last question is one we have considered in considerable detail for the standard account and found it lacking: first, because modular specification is not accompanied by modular reasoning (failure of monotonicity) and second, for the reasons discussed in section 5.5 (the potential folly of interpreting over-specification (magic) as under-specification (chaos)).

The second question, concerning appropriate operations, is also important. After all, there is no reason to suppose that these would be *exactly* those of Z, and for at least two reasons: first, because the semantics is different, and second, because the very *purpose* of the schema operators is generalised beyond specification in a framework which stresses refinement: it now includes issues in design and implementation.[13] The space of possibilities here is quite large, and the territory relatively unexplored. There remains, then, a number of theoretical and pragmatic questions which map out interesting avenues for future detailed investigation.

Exploration of specification combinators in other, or mixed, frameworks has an established history. Ward in [58] examines specification constructs in the Refinement Calculus, including conjunction and disjunction, though these are not monotonic. That work extends [42] which is possibly the earliest examination of connections between Z and the Refinement Calculus together with issues of schema algebra. More recently Paige [50] examined Z and other formalisms within the context of heterogeneous frameworks, motivated by method integration. Here, it would be possible to translate Z schemas into, for example, predicative specifications [32] accessing better monotonicity properties as a consequence. But the translations are at the level of atomic schemas, so any Z structure must, in general, be removed. Having said that, Ward [58] does provide a few specialised laws which maintain structure, which would extend to the heterogeneous framework of Paige: these include refining disjunctions directly to alternations and conjunctions directly to compositions, under strong sideconditions.

There are other approaches which consider program development directly from Z specifications, the most ambitious being ZRC [11, 9]. In this approach Z is given a WP-semantics (provided in [10]) equivalent to its (then) standard semantics (see [8]) and this is used to integrate specification with Refinement Calculus. The passage from Z specifications to specification statements induces preconditions in the standard way: as feasibility conditions. The use of schema operators is hampered (there are quite strong sideconditions on rules involving schema operators) because the Refinement Calculus solves the *frame-problem* by insisting that observations outside this frame do not change (this is a trivial consequence of the WP-semantics) and this does not sit well with component schemas which have overlapping or disjoint frames. Despite these problems the approach is well-developed and has much to recommend it: the approaches described in [37, 18] (which specifically address the disadvantages just listed, but doubtless have limitations of their own) can be carefully compared in the future with ZRC.

[13] Consider, for example, the role of sequencing in Refinement Calculus.

7. Acknowledgements

We would like to thank the New Zealand Foundation for Research, Science and Technology (references: UOWX0011 and UOWX0202) and the Royal Society of Great Britain, for financially supporting this research. Moshe Deutsch is supported by the British Council through an ORS award.
This work has been influenced in its development by too many people to name explicitly. However, special thanks for particularly important discussions and comments go to Rob Arthan, Steve Dunne, Lindsay Groves, Greg Reeve, David Streader, Ray Turner and Mark Utting.

References

[1] *APSEC 2003: 10th Asia-Pacific Software Engineering Conference, Chiangmai, Thailand, December 10-12, 2003, Proceedings*. IEEE Computer Society Press, December 2003. To appear.
[2] J. R. Abrial. *The B-Book*. Cambridge University Press, 1996.
[3] R. J. R. Back and J. von Wright. *Refinement Calculus: A Systematic Introduction*. Springer, 1998.
[4] R. Barden, S. Stepney, and D. Cooper. *Z in Practice*. Prentice Hall, 1994.
[5] D. Bjørner, C. A. R. Hoare, and H. Langmaack, editors. *VDM '90, VDM and Z – Formal Methods in Software Development, Third International Symposium of VDM Europe, Kiel, FRG, April 17-21, 1990, Proceedings*, volume 428 of *Lecture Notes in Computer Science*. Springer, 1990.
[6] C. Bolton, J. Davies, and J. C. P. Woodcock. On the Refinement and Simulation of Data Types and Processes. In K. Araki, A. Galloway, and K. Taguchi, editors, *Integrated Formal Methods (IFM '99)*. Springer, 1999.
[7] J. P. Bowen, S. Dunne, A. Galloway, and S. King, editors. *ZB 2000: Formal Specification and Development in Z and B, First International Conference of B and Z Users, York, UK, August 29 - September 2, 2000, Proceedings*, volume 1878 of *Lecture Notes in Computer Science*. Springer, 2000.
[8] S. Brien and J. Nicholls. Z base standard version 1.0. Technical report, University of Oxford, 1992.
[9] A. Cavalcanti. *A Refinement Calculus for Z*. PhD thesis, University of Oxford, 1997.
[10] A. Cavalcanti and J. C. P. Woodcock. A weakest precondition semantics for Z. *The Computer Journal*, 41(1):1–15, 1998.
[11] A. Cavalcanti and J. C. P. Woodcock. ZRC – a refinement calculus for Z. *Formal Aspects of Computing*, 10(3):267–289, 1998.
[12] D.Bert, J. P. Bowen, S. King, and M. Waldén, editors. *ZB 2003: Formal Specification and Development in Z and B, Third International Conference of B and Z Users, Turku, Finland, June 4-6, 2003, Proceedings*, volume 2651 of *Lecture Notes in Computer Science*. Springer, 2003.
[13] J. Derrick and E. Boiten. *Refinement in Z and Object-Z: Foundations and Advanced Applications*. Formal Approaches to Computing and Information Technology – FACIT. Springer, May 2001.
[14] M. Deutsch and M. C. Henson. An Analysis of Backward Simulation Data-Refinement for Partial Relation Semantics. In *APSEC 2003 [1]*, page tba.
[15] M. Deutsch and M. C. Henson. An Analysis of Forward Simulation Data Refinement. In *ZB 2003 [12]*, pages 148–167.
[16] M. Deutsch and M. C. Henson. An analysis of total correctness refinement models for partial relation semantics II. *Logic Journal of the IGPL*, 11(3):319–352, 2003.
[17] M. Deutsch and M. C. Henson. Four Theories for Backward Simulation Data-Refinement. In T. Muntean and K. Sere, editors, *RCS'03 – 2nd International Workshop on Refinement of Critical Systems: Methods, Tools and Developments*, Åbo Academi, Turku – Finland, June 2003.
[18] M. Deutsch, M. C. Henson, and B. Kajtazi. Modular Refinement in Novel Schema Calculi. In *APSEC 2003 [1]*, page tba.
[19] M. Deutsch, M. C. Henson, and S. Reeves. Operation Refinement and Monotonicity in the Schema Calculus. In *ZB 2003 [12]*, pages 103–126.
[20] M. Deutsch, M. C. Henson, and S. Reeves. Results on Formal Stepwise Design in Z. In P. Strooper and P. Muenchaisri, editors, *APSEC 2002: 9th Asia-Pacific Software Engineering Conference, Gold Coast, Queensland, Australia, December 4-6, 2003, Proceedings*, pages 33–42. IEEE Computer Society Press, December 2002.
[21] M. Deutsch, M. C. Henson, and S. Reeves. An analysis of total correctness refinement models for partial relation semantics I. *Logic Journal of the IGPL*, 11(3):287–317, 2003.
[22] M. Deutsch, M. C. Henson, and S. Reeves. Operation refinement and monotonicity in the schema calculus. *University of Essex, technical report CSM-381*, February 2003.
[23] E. W. Dijkstra. *A Discipline of Programming*. Prentice-Hall, 1976.
[24] A. Diller. *Z: An Introduction to Formal Methods*. J. Wiley and Sons, 2nd edition, 1994.
[25] S. Dunne. Private communication at *ZB 2003* [12]. Turku, Finland, June 2003.
[26] C. Fidge. Real-time refinement. In *FME '93 [66]*, pages 314–331.
[27] H. S. Goodman. From Z Specifications to Haskell Programs: A Three-Pronged Approach. In *Proceedings of the*

7th International Conference on: Putting into Practice Methods and Tools for Information System Design – Z Twenty Years on - What is its Future ?, pages 167–182, Nantes – France, October 1995.
[28] L. J. Groves. *Evolutionary Software Development in the Refinement Calculus*. PhD thesis, Victoria University, 2000.
[29] L. J. Groves. Refinement and the Z Schema Calculus. In *REFINE 2002: Refinement Workshop*. BCS FACS, July 2002.
[30] J. Grundy. *A Method of Program Refinement*. PhD thesis, University of Cambridge, 1993.
[31] I. Hayes. *Specification Case Studies*. Prentice Hall, 2nd edition, 1993.
[32] E. C. R. Hehner. *A Practical Theory of Programming*. Springer-Verlag, 1993.
[33] M. C. Henson and S. Reeves. Program Development and Specification Refinement in the Schema Calculus. In *ZB 2000 [7]*, pages 344–362.
[34] M. C. Henson and S. Reeves. Constructive Foundations For Z. In S. Kuru, M. U. Caglayan, and H. L. Akin, editors, *Proc. 12th International Symposium on Computer and Information Sciences: ISCIS XII*, pages 585–591. Bogazici University press, 1997.
[35] M. C. Henson and S. Reeves. New Foundations for Z. In J. Grundy, M. Schwenke, and T. Vickers, editors, *Proc. International Refinement Workshop and Formal Methods Pacific '98*, pages 165–179. Springer, 1998.
[36] M. C. Henson and S. Reeves. Investigating Z. *Logic and Computation*, 10(1):43–73, 2000.
[37] M. C. Henson and S. Reeves. A logic for schema-based program development. *Formal Aspects of Computing*, 15(1):84–99, 2003.
[38] J. Jacky. Formal specification of control software for a radiation therapy machine. *Radiation Oncology Department, University of Washington, technical report 94-07-01*, 1994.
[39] M. Johnson and P. Sanders. From Z Specifications To Functional Implementations. In J. E. Nicholls, editor, *Z User Workshop – Proceedings of the 4th Annual Z User Meeting*, pages 86–112. Springer-Verlag, 1990.
[40] C. B. Jones. *Software Development: A Rigorous Approach*. Prentice-Hall International, 1980.
[41] C. B. Jones. *Systematic Software Development using VDM*. Prentice-Hall International, 2nd edition, 1990.
[42] S. King. Z and the Refinement Calculus. In *VDM '90 [5]*, pages 164–188.
[43] S. King and I. H. Sørensen. From Specification, Through Design to Code: A Case Study in Refinement. In P. N. Scharbach, editor, *Formal Methods: Theory and Practice*, pages 103–137. BSP Professional books, 1989.
[44] B. P. Mahony. The least conjunctive refinement and promotion in the refinement calculus. *Formal Aspects of Computing*, 11:75–105, 1999.
[45] P. Martin-Löf. Constructive mathematics and computer programming. In *Logic, Methodology and Philosophy of Science VI*, pages 153–175. North Holland, 1982.
[46] S. H. Mirian-Hosseinabadi and R. Turner. Constructive Z. *Logic and Computation*, 8(1):49–70, 1998.
[47] C. C. Morgan. The specification statement. *ACM Transactions on Programming Languages and Systems*, 10:403–419, 1988.
[48] C. C. Morgan. *Programming from Specifications*. Prentice Hall International, 2nd edition, 1994.
[49] C. C. Morgan and B. Surfin. Specification of the unix filing system. *IEEE Transactions on Software Engineering*, 10(2):128–142, 1984.
[50] R. F. Paige. Heterogeneous notions for pure formal method integration. *Formal Aspects of Computing*, 10(3):233–242, 1998.
[51] B. Potter, J. Sinclair, and D. Till. *An Introduction to Formal Specification and Z*. Prentice Hall, 2nd edition, 1996.
[52] S. Prehn and W. J. Toetenel, editors. *VDM '91 – Formal Software Development, 4th International Symposium of VDM Europe, Noordwijkerhout, The Netherlands, October 21-25, 1991, Proceedings, Volume 2: Tutorials*, volume 552 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.
[53] G. Smith. A Formal Specification of Signalling System No. 7: Telephone User Part. In *Proceedings of the Singapore International Conference on Networks 1989 (SICON '89)*. IEEE Computer Press, July 1989.
[54] J. M. Spivey. *The Z Notation: A Reference Manual*. Prentice Hall, 2nd edition, 1992.
[55] B. Strulo. How Firing Conditions Help Inheritance. In J. P. Bowen and M. G. Hinchey, editors, *ZUM '95: The Z Formal Specification Notation*, volume 967 of *Lecture Notes in Computer Science*, pages 264–275. Springer-Verlag, 1995.
[56] I. Toyn, editor. *Z Notation: Final Committee Draft, CD 13568.2*. Z Standards Panel, 1999.
[57] M. Utting. Private communication. Department of Computer Science, University of Waikato, Hamilton, New Zealand, June 2002.
[58] N. Ward. Adding Specification Constructors to the Refinement Calculus. In *FME '93 [66]*, pages 652–670.
[59] J. C. P. Woodcock. An Introduction to Refinement in Z. In *VDM '91 [52]*, pages 96–117.
[60] J. C. P. Woodcock. The Refinement Calculus. In *VDM '91 [52]*, pages 80–95.
[61] J. C. P. Woodcock. Two Refinement Case Studies. In *VDM '91 [52]*, pages 118–140.
[62] J. C. P. Woodcock. Calculating properties of Z specifications. *ACM SIGSOFT Software Engineering Notes*, 14(5):43–54, 1989.
[63] J. C. P. Woodcock. Implementing Promoted Operations in Z. In C. B. Jones, R. C. Shaw, and T. Denvir, editors, *5th Refinement Workshop*, Workshops in Computing, pages 367–378. Springer-Verlag, 1992.
[64] J. C. P. Woodcock. The rudiments of algorithm refinement. *The Computer Journal*, 35(5):441–450, 1992.
[65] J. C. P. Woodcock and J. Davies. *Using Z: Specification, Refinement and Proof*. Prentice Hall, 1996.

[66] J. C. P. Woodcock and P. G. Larsen, editors. *FME '93: Industrial-Strength Formal Methods, 1st International Symposium of Formal Methods Europe, Odense, Denmark, April 19-23, 1993, Proceedings*, volume 670 of *Lecture Notes in Computer Science*. Springer, 1993.

[67] J. B. Wordsworth. *Software Development with Z – A Practical Approach to Formal Methods in Software Engineering*. Internalional Computer Science Series. Addison-Wesley, 1992.

A. Specification logic - a synopsis

In this appendix, we will revise a little Z logic, settling our notational conventions in the process. The reader may wish to consult [36] and [21] for a more leisurely treatment of our notational and meta-notational conventions.
Our analysis takes place in $\mathcal{Z}_C^\perp$ [21], which is a simple *conservative extension* of the "Church-style" version of the Z-logic, $\mathcal{Z}_C$ [36], which incorporates $\perp$ into its types. This provides a satisfactory logical account of the schema calculus of Z as it is normally understood, and permits a formalisation of the various relational completion operators we consider.

A.1. Schemas

$\mathcal{Z}_C^\perp$ is a typed theory in which the types of higher-order logic are extended with *schema types* whose values are unordered, label-indexed tuples called *bindings*. For example, if the T_i are types and the $\mathtt{z}_i$ are *observations* (constants) then:

$$[\cdots \mathtt{z}_i : T_i \cdots]$$

is a (schema) type. Values of this type are bindings, of the form:

$$\langle\!| \cdots \mathtt{z}_i \Rrightarrow t_i^{T_i} \cdots |\!\rangle$$

where $t_i^{T_i}$ means that the term t_i has type T_i. *Binding selection*, written $t.\mathtt{x}$, is axiomatised so that, for example:

$$\langle\!| \mathtt{x} \Rrightarrow 2, \mathtt{y} \Rrightarrow 3 |\!\rangle.\mathtt{x} = 2$$

Selection generalises so that $t.P$ denotes the predicate P in which each observation $\mathtt{x}$ is replaced by $t.\mathtt{x}$. *Filtered* bindings play a major role in the schema calculus. Such terms have the form $t \upharpoonright T$ and are axiomatised so that, for example:

$$\langle\!| \mathtt{x} \Rrightarrow 2, \mathtt{y} \Rrightarrow 3 |\!\rangle \upharpoonright [\mathtt{x} : \mathbb{N}] = \langle\!| \mathtt{x} \Rrightarrow 2 |\!\rangle$$

The symbols $\preceq$, $\wedge$, $\vee$ and $-$ denote the *schema subtype* relation, and the operations of *schema type intersection* and (compatible) *schema type union* and *schema type subtraction*. We let U (with diacriticals when necessary) range over schema expressions. When U is an operation schema, these are sets of bindings linking, as usual, before observations with after observations. This captures the informal account to be found in the literature (*e.g.* [24], [65]). We can always write the type of such operation schemas as $\mathbb{P}(T^{in} \vee T^{out'})$ where T^{in} is the type of the "before" sub-binding (state) and $T^{out'}$ is the type of the "after" sub-binding. We also permit *binding concatenation*, written $t_0 \star t_1$, when the alphabets of t_0 and t_1 are disjoint. We lift this operation to sets (of appropriate type), with obvious introduction and elimination rules, by means of:

Definition A.1.

$$C_0 \star C_1 =_{df} \{z_0 \star z_1 \mid z_0 \in C_0 \wedge z_1 \in C_1\}$$

The same restriction obviously applies here: the types of the sets involved must be disjoint.
We introduce two notational conventions in order to avoid the repeated use of filtering in the context of membership and equality propositions.

Definition A.2. Let $T_1 \preceq T_0$.

$$t^{T_0} \dot{\in} C^{\mathbb{P} T_1} =_{df} t \upharpoonright T_1 \in C$$

Definition A.3. Let $T_1 \preceq T_0$ or $T_0 \preceq T_1$.

$$t_0^{T_0} \doteq t_1^{T_1} =_{df} t_0 \upharpoonright (T_0 \wedge T_1) = t_1 \upharpoonright (T_0 \wedge T_1)$$

$\mathcal{Z}_C^\perp$ includes distinguished terms which are needed in the approach taken in [65] for completing relations. Specifically: the types include terms $\perp^T$ for every type T. There are, additionally, a number of axioms which ensure that all the new $\perp^T$ values interact properly, for example:

Axiom A.1.

$$\overline{\perp^{[\mathtt{z}_0:\mathtt{T}_0 \cdots \mathtt{z}_n:\mathtt{T}_n]} = \langle\!| \mathtt{z}_0 \Rrightarrow \perp^{T_0} \cdots \mathtt{z}_n \Rrightarrow \perp^{T_n} |\!\rangle}$$

□

In other words, $\bot^{[\mathtt{z_0}:\mathtt{T_0}\cdots\mathtt{z_n}:\mathtt{T_n}]}.\mathtt{z}_i = \bot^{T_i}$ $(0 \leq i \leq n)$. Note that this is the *only* axiom concerning distinguished bindings, hence, binding construction is *non-strict* with respect to the $\bot^T$ values. We show in [21] that $\mathcal{Z}_C^\bot$ is a *conservative extension* of the original Z logic $\mathcal{Z}_C$.

A.2. The schema calculus in $\mathcal{Z}_C^\bot$

The definitional extension $\mathcal{Z}_C^\bot$, of $\mathcal{Z}_C$, which introduces schemas as sets of bindings and the various operators of the schema calculus, is undertaken as usual (see [36]) but, in $\mathcal{Z}_C^\bot$, the carrier sets of the types must be adjusted to form what we call the *natural carrier sets* which are those sets of elements of types that *explicitly exclude* the $\bot^T$ values:

Definition A.4. *Natural carriers* for each type are defined by closing: $\mathbb{N} =_{df} \{z^{\mathbb{N}} \mid z \neq \bot^{\mathbb{N}}\}$ under the operations of cartesian product, powerset and schema set.[14]

Definition A.5 (Semantics for atomic schemas). $[\![[T \mid P]]\!]_0 =_{df} \{z \in T \mid z.P\}$

Note that this definition[15] draws bindings from the *natural carrier* of the type T. As a consequence, writing $t(\bot)$ for a binding satisfying $t.\mathbf{x} = \bot$ for some observation $\mathbf{x}$, we have:

Lemma A.1.

$$\frac{t(\bot) \in U}{false}$$

□

We will also need the *extended carriers*. These are defined for all types as follows:

Definition A.6.

$$T_\bot =_{df} T \cup \{\bot^T\}$$

In a similar manner to [36], we now define four operations in $\mathcal{Z}_C^\bot$, which pave the way for the interpretation of the schema algebra in $\mathcal{Z}_C^\bot$. They are polymorphic: the type of their instances is determined by the particular sets to which they are applied. Note that we use the meta-variable C to range over sets (terms of type $\mathbb{P}\, T$). T denotes the set $T^{in} \star T^{out'}$ and $T^\star$ denotes the set $T_\bot^{in} \star T_\bot^{out'}$.

Definition A.7. Let z be a fresh variable in all the four definitions below.

Conjunction and disjunction. Let T^{in} be $T_0^{in} \vee T_1^{in}$ and $T^{out'}$ be $T_0^{out'} \vee T_1^{out'}$.

$$(i) \quad C_0^{\mathbb{P}\, T_0} \wedge C_1^{\mathbb{P}\, T_1} \quad =_{df} \quad \{z \in T^\star \mid z \mathrel{\dot\in} C_0 \wedge z \mathrel{\dot\in} C_1\}$$

$$(ii) \quad C_0^{\mathbb{P}\, T_0} \vee C_1^{\mathbb{P}\, T_1} \quad =_{df} \quad \{z \in T^\star \mid z \mathrel{\dot\in} C_0 \vee z \mathrel{\dot\in} C_1\}$$

Composition.
We will deal with instances of *composition* where the expression $C_0 \fatsemi C_1$ has the type $\mathbb{P}(T_0^{in} \vee T_1^{out'})$ and where C_0 is of type $\mathbb{P}(T_0^{in} \vee T_0^{out'})$, C_1 is of type $\mathbb{P}(T_1^{in} \vee T_1^{out'})$ and $T_0^{out} = T_1^{in}$. With all this in place we can omit the type superscripts when analysing composition in the paper.

$$(iii) \quad C_0^{\mathbb{P}(T_0^{in} \vee T_0^{out'})} \fatsemi C_1^{\mathbb{P}(T_1^{in} \vee T_1^{out'})} =_{df} \{z_0 \star z_1' \in T_{0_\bot}^{in} \star T_{1_\bot}^{out'} \mid \exists\, y^{T_0^{out}} \bullet z_0 \star y' \in C_0 \wedge y \star z_1' \in C_1\}$$

Existential quantification. Let $T_\mathtt{z} =_{df} [\mathtt{z} : T^\mathtt{z}]$. If C has the type $\mathbb{P}\, T$ and $T_\mathtt{z} \preceq T$, then an expression of the form $\exists \mathtt{z} : T^\mathtt{z} \bullet C$ will always have the type $\mathbb{P}(T - T_\mathtt{z})$; thus, we can omit the type superscripts when analysing existential hiding in the paper.

$$(iv) \quad \exists \mathtt{z} : T^\mathtt{z} \bullet C^{\mathbb{P}\, T} =_{df} \{x \in (T - T_\mathtt{z})^\star \mid \exists\, y \in T \bullet y \in C \wedge x = y \upharpoonright (T - T_\mathtt{z})\}$$

With all these in place, we can now interpret schema expressions in $\mathcal{Z}_C^\bot$:

[14] The notational ambiguity does not introduce a problem, since only a set can appear in a term or proposition, and only a type can appear as a superscript.
[15] We write this interpretation as $[\![\_]\!]_0$ so as to distinguish it from the ones used in section 5.1 in the main body of the paper.

Definition A.8.

$$\begin{array}{lll} (i) & [\![U_0 \wedge U_1]\!]_0 & =_{df} \ [\![U_0]\!]_0 \wedge [\![U_1]\!]_0 \\ (ii) & [\![U_0 \vee U_1]\!]_0 & =_{df} \ [\![U_0]\!]_0 \vee [\![U_1]\!]_0 \\ (iii) & [\![U_0 \fatsemi U_1]\!]_0 & =_{df} \ [\![U_0]\!]_0 \fatsemi [\![U_1]\!]_0 \\ (iv) & [\![\exists \mathtt{z} : T^{\mathtt{z}} \bullet U]\!]_0 & =_{df} \ \exists \mathtt{z} : T^{\mathtt{z}} \bullet [\![U]\!]_0 \end{array}$$

We then immediately get the following derived rules for the various schema operators in $\mathcal{Z}_C^{\perp}$:

Proposition A.1.
Schema conjunction. Let $i \in 2$.

$$\frac{t \mathrel{\dot\in} U_0 \quad t \mathrel{\dot\in} U_1}{t \in U_0 \wedge U_1} \ (U^+_{\wedge}) \qquad \frac{t \in U_0 \wedge U_1}{t \mathrel{\dot\in} U_i} \ (U^-_{\wedge_i}) \qquad \frac{t \in U_0 \wedge U_1}{t \in T^\star} \ (U^-_{\wedge_2})$$

Schema disjunction. Let $i \in 2$.

$$\frac{t \in T^\star \quad t \mathrel{\dot\in} U_i}{t \in U_0 \vee U_1} \ (U^+_{\vee_i}) \qquad \frac{t \in U_0 \vee U_1 \quad t \mathrel{\dot\in} U_0 \vdash P \quad t \mathrel{\dot\in} U_1 \vdash P}{P} \ (U^-_{\vee_0}) \qquad \frac{t \in U_0 \vee U_1}{t \in T^\star} \ (U^-_{\vee_1})$$

Schema composition.

$$\frac{t_0 \star y' \in U_0 \quad y \star t_1' \in U_1}{t_0 \star t_1' \in U_0 \fatsemi U_1} \ (U^+_{\fatsemi}) \qquad \frac{t_0 \star t_1' \in U_0 \fatsemi U_1 \quad t_0 \star y' \in U_0, y \star t_1' \in U_1 \vdash P}{P} \ (U^-_{\fatsemi_0}) \qquad \frac{t_0 \star t_1' \in U_0 \fatsemi U_1}{t_0 \star t_1' \in T^{in}_{0_\perp} \star T^{out'}_{1_\perp}} \ (U^-_{\fatsemi_1})$$

Schema existential hiding.

$$\frac{t \in U}{t \mathrel{\dot\in} \exists \mathtt{z} : T^{\mathtt{z}} \bullet U} \ (U^+_{\exists}) \qquad \frac{t \in \exists \mathtt{z} : T^{\mathtt{z}} \bullet U \quad y \in U, y \doteq t \vdash P}{P} \ (U^-_{\exists_0}) \qquad \frac{t \in \exists \mathtt{z} : T^{\mathtt{z}} \bullet U}{t \in (T - T_{\mathtt{z}})^\star} \ (U^-_{\exists_1})$$

The usual sideconditions apply to the eigenvariable y in both $(U^-_{\fatsemi_0})$ and $(U^-_{\exists_0})$. □

Proposition A.2 (Composition logic). Notice that the above rules for schema composition constitute a major simplification to the ones provided in [36]. We require a simplification of the equational logic for schema composition.
Let $T^{in} = [\mathtt{x} : T]$ and $T^{out'} = [\mathtt{x}' : T]$ then the following equation is derivable for schema composition:

$$[\mathtt{x}, \mathtt{x}' : T \mid P_0] \fatsemi [\mathtt{x}, \mathtt{x}' : T \mid P_1] = [\mathtt{x}, \mathtt{x}' : T \mid \exists\, v^{T^{in}} \bullet P_0[\mathtt{x}'/v'.\mathtt{x}'] \wedge P_1[\mathtt{x}/v.\mathtt{x}]]$$

□

The schema calculus in $\mathcal{Z}_C^{\perp}$ preserves the meaning of the schema calculus in $\mathcal{Z}_C$. This is established by induction over the structure of schema expressions, which shows that the $\mathcal{Z}_C^{\perp}$ natural carriers and the $\mathcal{Z}_C$ carrier sets lead to equivalent schema calculi. Hence, the $\mathcal{Z}_C^{\perp}$ schema calculus is *hereditarily $\perp$-free*.

A.3. Preconditions

We can formalise the idea of the *precondition* of an operation schema, that is, the domain of the relation between the before-states and after-states that the schema denotes.

Definition A.9. Let $T^{in} \preceq V$.

$$Pre\ U\ x^V =_{df} \exists z \in U \bullet x \doteq z$$

Proposition A.3. Let y be a fresh variable, then the following introduction and elimination rules are immediately derivable for preconditions:

$$\frac{t_0 \in U \quad t_0 \doteq t_1}{Pre\ U\ t_1} \qquad \frac{Pre\ U\ t \quad y \in U, y \doteq t \vdash P}{P}$$

□

In general the precondition of an operation schema will not be the whole of T^{in}. In this sense operation schemas denote partial relations.
A key to reasoning about the monotonicity properties of the various schema calculus operators is the necessity of reasoning about the precondition of operations defined by schema operations. This topic has been investigated informally in, for example, [62] and [67] (for conjoined and disjoined operation schemas); and formally, for the schema calculus generally, in [19].
We now provide complete precondition theories for schema conjunction, schema disjunction, schema existential hiding and schema composition. We shall not provide the proofs for the various results; should the reader be interested, a complete account for these is provided in [22].

A.3.1. The precondition for conjunction

In general, the precondition of a conjunction of operations is not the conjunction of the preconditions of the individual components [62]. This is a direct consequence of the underlying "postcondition only" approach Z takes.
In fact, we can be more precise: the usual form of a conjunction introduction rule fails, whereas the elimination rules hold. An example, which embodies this result, can be found in [62]. This can be remedied using a strong syntactic sidecondition, for example insisting that the alphabets of the operations are disjoint (see, for example, [29] and [67, p.214]).

Proposition A.4. Let $i \in 2$, then the following elimination rules are derivable for the precondition of conjoined schemas:

$$\frac{Pre\,(U_0 \wedge U_1)\,t}{Pre\;U_i\;t}\;(Pre^{-}_{\wedge_i})$$

□

A.3.2. The precondition for disjunction

The precondition of schema disjunction is better behaved. This is due to the fact that existential quantification is disjunctive; that is, fully distributes over disjunction (see [62], [65, p.210] and [67, p.125]). Hence, we get the following results:

Proposition A.5. Let $i \in 2$, then the following introduction and elimination rules for the precondition of the disjunction of schemas are derivable:

$$\frac{Pre\;U_i\;t}{Pre\,(U_0 \vee U_1)\,t}\;(Pre^{+}_{\vee_i}) \qquad \frac{Pre\,(U_0 \vee U_1)\,t \quad Pre\;U_0\;t \vdash P \quad Pre\;U_1\;t \vdash P}{P}\;(Pre^{-}_{\vee})$$

□

With these in place, we can easily prove the full distributivity of the precondition over disjunction (this is also stated in [64]).

Theorem A.1.

$$Pre\,(U_0 \vee U_1)\,t \Leftrightarrow Pre\;U_0\;t \vee Pre\;U_1\;t$$

□

A.3.3. The precondition for existential quantification

The following rules are derivable:

Proposition A.6.

$$\frac{Pre\;U\;t}{Pre\,(\exists\,\mathtt{z} : T^{\mathtt{z}} \bullet U)\,t}\;(Pre^{+}_{\exists}) \qquad \frac{Pre\,(\exists\,\mathtt{z} : T^{\mathtt{z}} \bullet U)\,t \quad Pre\;U\;y, y \doteq t \vdash P}{P}\;(Pre^{-}_{\exists})$$

Note that the usual sideconditions apply to the eigenvariable y. □

A.3.4. The precondition for composition

The following introduction and elimination rules for the precondition of composed operation schemas are derivable:

Proposition A.7.

$$\frac{t_0 \star t_1' \in U_0 \quad Pre\;U_1\;t_1}{Pre\,(U_0 \fatsemi U_1)\,t_0}\;(Pre^{+}_{\fatsemi}) \qquad \frac{Pre\,(U_0 \fatsemi U_1)\,t_0 \quad Pre\;U_1\;y, t_0 \star y' \in U_0 \vdash P}{P}\;(Pre^{-}_{\fatsemi})$$

The usual sideconditions apply to the eigenvariable y. □

Lemma A.2. The following additional rule is derivable for the precondition of composition:

$$\frac{Pre\,(U_0 \fatsemi U_1)\,t_0}{Pre\;U_0\;t_0}$$

□

A.4. Relational completion

In this section, we review three relational completion operators discussed in [21] and [20].
We begin by defining the non-strict-lifted-totalisation in line with the intentions described in [65], chapter 16.

Definition A.10 (Non-strict-lifted-totalisation).

$$\dot{U} =_{df} \{z_0 \star z_1' \in T^\star \mid Pre\;U\;z_0 \Rightarrow z_0 \star z_1' \in U\}$$

Then the following introduction and elimination rules are derivable:

Proposition A.8.

$$\frac{t_0 \star t_1' \in T^\star \quad Pre\;U\;t_0 \vdash t_0 \star t_1' \in U}{t_0 \star t_1' \in \dot{U}}\;(\bullet^+) \qquad \frac{t_0 \star t_1' \in \dot{U} \quad Pre\;U\;t_0}{t_0 \star t_1' \in U}\;(\bullet_0^-) \qquad \frac{t_0 \star t_1' \in \dot{U}}{t_0 \star t_1' \in T^\star}\;(\bullet_1^-)$$

□

Lemma A.3. The following extra rules are derivable for lifted-totalised sets:

$$\frac{}{U \subseteq \dot{U}}\;(i) \qquad \frac{}{\bot \in \dot{U}}\;(ii) \qquad \frac{t' \in T_\bot^{out'}}{\bot \star t' \in \dot{U}}\;(iii) \qquad \frac{\neg\,Pre\;U\;t \quad t \in T_\bot^{in}}{t \star \bot' \in \dot{U}}\;(iv) \qquad \frac{\neg Pre\;U\;t_0 \quad t_0 \in T_\bot^{in} \quad t_1' \in T_\bot^{out'}}{t_0 \star t_1' \in \dot{U}}\;(v)$$

□

Another relational completion lifts and totalises the relation, but is strict with respect to abortive behaviour: $\bot$ maps only to $\bot$.

Definition A.11 (Strict-lifted-totalisation).

$$\dot{\ominus U} =_{df} \{z_0 \star z_1' \in T^\star \mid Pre\;U\;z_0 \Rightarrow z_0 \star z_1' \in U \wedge z_0 = \bot \Rightarrow z_1' = \bot'\}$$

Our final relational completion is the *abortive-lifted-totalisation*: $\bot$ and all before-states outside the precondition, map only to $\bot$.

Definition A.12 (Abortive-lifted-totalisation).

$$\dot{\Box U} =_{df} \{z_0 \star z_1' \in T^\star \mid z_0 \star z_1' \in U \vee (\neg\,Pre\;U\;z_0 \wedge z_1' = \bot')\}$$